Aalto University
School of Electrical Engineering
Department of Communications and Networking

XUEYUN LI

Performance Evaluation of three Data Access Control Schemes for Cloud Computing

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Espoo,

Thesis Instructor: Docent & Visiting Professor Zheng Yan (D.Sc Tech)

Thesis Supervisor: Professor Raimo Kantola (D.Sc Tech)

AALTO UNIVERSITY Abstract of the

SCHOOL OF ELECTRICAL ENGINEERING Master's Thesis

Author: Xueyun LI

Title: Performance Evaluation of three Data Access Control Schemes for Cloud Computing

Date: 20.04.2015 **Language**: English **Number of Pages**: 8+83

Department: Department of Communications and Networking

Professorship: Networking Technology **Code**: S-38

Supervisor: Professor Raimo Kantola

Instructor: Professor Zheng Yan

Cloud services are flourishing recently, both among computer users and business enterprises. They deliver remote, on-demand, convenient services for data storage, access and processing. While embracing the benefits brought by various cloud services, the consumers are faced with data disclosure, privacy leaks and malicious attacks. Therefore, it is important to use strong access control policies to maintain the security and confidentiality of the data stored in the cloud.

This thesis studies the performance of three existing security schemes proposed for cloud data access control on the basis of trust and reputation. We implement the three schemes and conduct computation complexity analysis, security analysis and performance evaluation. This thesis introduces the implementation of a number of cryptographic algorithms applied in the above data access control schemes, including Proxy Re-encryption (PRE) and Ciphertext-Policy Attribute Based Encryption (CP-ABE), reputation generation and secure data transmission over Secure Socket Layer (SSL). We summarize the evaluation results and compare the performances in the aspects of computation and communication costs, flexibility, scalability and feasibility of practical usage. Pros and cons, as well as suitable application scenarios of the three schemes are further discussed.

Keywords: Cloud Computing, Access Control, Proxy Re-Encryption, Attribute Based Encryption, Trust Management, Reputation Systems, Performance Evaluation, Computational Efficiency

Acknowledgement

I wish to thank, first, my supervisor Professor Raimo Kantola for his continuous support, understanding and expertise which added considerably to my graduate experience.

My sincerest thanks go to my instructor Professor Zheng Yan, for her novel ideas, rich experience and insightful comments. Guided by her enthusiasm, motivation and immense knowledge, I learned the methods and meticulous attitude toward research work. She also helped me with my writing techniques and thesis presentations during the whole thesis working period. This thesis would not have been possible without her help and encouragement.

I also owe my deep gratitude to all the lecturers and assistants who provided me substantial knowledge and guidance during my studies in Aalto University. I also like to thank my dearest friends who are being supportive no matter in my studies or everyday life during my time in Finland.

Last but not least, I would like to thank my parents, Jianqiu Li and Haiping Xiao, for loving me and supporting me anytime and anywhere.

Otaniemi, Espoo,

Xueyun LI

List of Tables:

List of Figures:

Abbreviations

PRE	Proxy Re-Encryption
ABE	Attribute Based Encryption
KP-ABE	Key Policy-Attribute Based Encryption
CP-ABE	Ciphertext Policy-Attribute Based Encryption
CSP	Cloud Service Provider
RC	Reputation Center
PK	Public Key
SK	Secret Key
MK	Master Key

Table of Content:

Chapter 1 Introduction

1.1 Background and motivation

Cloud computing, as an emerging computing paradigm, is blooming in both academia and industry. It is defined by the U.S. National Institute of Standards and Technology (NIST) as a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management efforts [1]. Cloud computing enables various cloud services to provide infrastructure, platform and software for ubiquitous communications and data access. For example, some popular cloud services, such as Google Drive, iCloud and Dropbox, have freed us from hardware restraints and ensured that the data we need are available at any time and in any place. However, while embracing the services and benefits brought by cloud computing, we are also facing the problems of data disclosure, privacy leaks and malicious attacks.

Since cloud services, especially cloud storage and data management, are gaining special attention from computer users and business enterprises, data security is becoming a major hurdle to widespread adoption of cloud technologies. For example, user's personal data were reported to be leaked from Google and Apple applications. And two data breaches took place at Oregon Health and Science University due to inappropriate handling and storage of unencrypted patient medical records in the cloud. Currently, access control in the cloud environment is typically provided by techniques such as VLANs and firewalls, which are originally designed to support enterprise IT systems [2]. These techniques are not suitable for the multi-tenant and dynamic nature of cloud, thus they could lead to major data breaches.

In order to meet the security and privacy requirements for data access control in cloud services, a number of solutions have been proposed, such as authorization [3][4], trust and reputation management [5][6][7][8], and data encryption [9][10][11][12][13][14]. Among these techniques, data encryption is considered a secure way to keep data confidentiality, because it depends less on the trust of cloud servers than other solutions. In addition, before the access control schemes are applied in practical systems, they should be implemented and evaluated with regard to meeting the requirements on security, flexibility, and computational efficiency. Based on the three existing security schemes proposed for cloud data access

control on the basis of trust and reputation [5], we conduct performance evaluation based on complexity analysis, security analysis and scheme implementation in order to provide their pros and cons regarding feasibility of practical usage.

1.2 Objectives of the thesis

The objective of this thesis is to implement and evaluate three different schemes for secure data access control in cloud computing. Through the evaluation and comparison, we discuss the pros and cons of each scheme for practical usage. Concretely, we aim to fulfill the following research tasks in order to achieve the objectives of the thesis:

Scheme implementation: This thesis work firstly intends to implement the three proposed scheme in terms of data access control in cloud computing. The scheme implementation consists of developing a number of cryptographic algorithms, reputation evaluation algorithms, trust level (TL) assessment algorithms, and secure communications. All the main functionalities of the above algorithms and protocols can be integrated to form a complete scheme and system implementation, as well as being called separately to provide independent operations related to the schemes.

Performance test: The performance of the proposed schemes is tested and analyzed under a certain environment based on the implantation of the above algorithms, and the test results are reported in details. The test results are deeply analyzed regarding different entities and the system as a whole.

Performance analysis, evaluation and comparison: Upon the implementation and performance test, computational complexity, data confidentiality, scalability and flexibility are further analyzed and compared.

Further discussion: Upon the implementations and performance evaluation, we further provide the pros and cons of the three access control schemes, as well as discuss their use scenarios. Additionally, in order to make the performance evaluation and comparison more thorough, we hope to come up with additional ideas and discussions towards further improvement and optimization.

1.3 Thesis structure

The thesis is organized as follows. Chapter 2 gives a review of related work of trust and reputation management, data access control methods and privacy issues in cloud computing. Chapter 3 introduces the cryptographic basics, and the design of three schemes for data access control in cloud computing based on trust and reputation. Chapter 4 describes the design of system implementation, which includes the implementation of Proxy Re-Encryption (PRE), Ciphertext Policy-Attribute Based Encryption (CP-ABE), secure transmission, and reputation/trust evaluation. Chapter 5 reports the results of performance evaluation and comparison, and further discusses the pros and cons of the three data access control schemes. Finally, the conclusions of the whole thesis with a proposal of future work are given in the last chapter.

1.4 Contributions

The contributions of the thesis can be summarized as follows:

1) Independently implemented three cloud data access control schemes based on the trust and reputation;
2) Conducted performance evaluation of the three schemes based on implementation, with regard to computation and communication cost, flexibility, scalability and feasibility of practical usage. Comparison was conducted among the three schemes and with other existing related schemes;
3) Further discussed the pros and cons of the three schemes based on performance comparison.

While writing this thesis, I co-authored several conference and journal papers in the subject of this thesis. My contributions for those publications enabled the scheme implementations and evaluations, which are relevant to this thesis. The publications and detailed contributions are presented in Appendix.

Chapter 2 Related Work

This chapter gives a brief introduction to several solutions for data access control, and surveys the methods of privacy preserving in cloud computing.

2.1 Trust and reputation management in cloud computing

Trust and reputation are mentioned in many literature sources, and they usually have no unique definitions because of their appearance in different research backgrounds. For instance, Hussain provided an overview on trust and reputation definitions, and indicated that the current notions of trust and reputation need to be further formally defined [15]. Dasgupta defined trust as "the expectation of one person about the actions of others that affects the first person's choice, when an action must be taken before the actions of others are known" [16]. Rahman defined reputation as "an expectation about an agent's behavior based on information or observations of its past behavior" [17]. As the research of trust and reputation grows fast in both theoretical foundations and real-world deployment, a number of e-commerce and cloud service companies apply them in ranking their products and suppliers, as well as building their recommendation systems. In cloud computing, trust and reputation are capitalized on designing mechanisms for data access control or recommendation systems, and are regarded as effective incentives to form a healthy and trustworthy network of participants who may have no prior knowledge of each other.

2.1.1 Trust management

Trust management is playing a critical role in increasing reliability, usability and security in cloud computing. Sato proposed a trust management model by dividing the model into two layers, namely the internal trust layer and contracted trust layer [18]. The internal trust layer is based on the Trusted Platform Module (TPM), which is an international standard for a secure cryptoprocessor. It handles strict security operations, such as key generation, key management and private data modification. The contracted trust layer is defined as the trust determined by certain contracts between service providers and users. A contract defines the trust and security level of a service provider, and the reliance between the provider and its users. Although there can be different trust levels in the negotiation on a contract, this model does not give specific factors for evaluating the trust levels. And it requires resigning the

contract whenever there is a service demand or security requirement update. Yao proposed an accountability-based system to achieve Trustworthy Service Oriented Architecture (TSOA) [19]. The trust and accountability management also depends on predefined and mutually agreed policies, called Service Level Agreement (SLA). It monitors each participant's behavior, and determines which participant is accountable according to the SLA. Pawar proposed a trust model to support the trustworthiness verification of cloud infrastructure providers [20]. The trust levels are calculated based on a feedback model in terms of trust, distrust and uncertainty. This approach also depends on a predefined SLA.

Prajapati proposed a trust management model for Software as a service (SaaS) in cloud environment [21]. In this model, trust is evaluated through *Trust* which indicates direct relations between participants, and *Recommended Trust* which is based on recommendations from other participants. However, this model assumes that all the nodes are honest, and is not resilient to malicious feedback or attacks. Habib proposed a multi-faceted Trust Management (TM) system to help users choose trustworthy cloud service providers [22]. The system evaluates trust levels through comprehensive attributes of Quality of Service (QoS), such as security, compliance, availability and response time. The TM system is centralized by registering all the cloud service providers in the Registration Manager (RM), and provides a Trust Manager (TMg) to allow service users to specify their requirements and feedback.

Yan proposed a Trust assessment model which enables both trust assessment between CSPs and users to help users choose trustworthy CSPs, and also the trust assessment between users to allow access right delegation [23]. This Trust assessment model is designed for cloud computing and mobile social networks. It takes mobile services (e.g., mobile calls, instant messages, pervasive interactions, etc.), weight parameters, priority levels and a punishment factor into account for participant's trust level assessment.

Table I provides the comparison of Trust management models proposed in [23][18][22], according to the following properties.

Table I Comparison of trust management model

	Sato [18]	Habib	Yan [23]
Factors for trust assessment	Not specified	QoS	Weight parameter; Priority level; Punishment factor
Enable trust assessment between CSPs and users	√	√	√
Enable trust assessment between users	-	-	√
Trust management through policy	√	-	-
Trust management through feedback	-	√	√
Enable attack resistance	-	√	-

Factors for trust assessment: Factors for trust assessment determine what properties and issues should be considered in trust evaluation and management. The factors depend on the application context or the design purposes. For example, Habib chooses QoS, and Yan selects performance and weight of behavior in mobile social network as evaluation parameters.

Enable trust assessment between CSPs and Users: It determines the purpose of Trust management system. Enabling trust assessment between CSPs and users is designed to help users choose trustworthy CSPs, and encourage CSPs to have better performance.

Enable trust assessment between users: It is designed for enabling data access right delegation between cloud service users.

Trust management through policy/feedback: This determines the technique used for trust assessment and management. For example, Sato monitors and manages the trust levels by signing contracts, while Habib evaluates and manages trust levels through feedback and recommendations.

Enable attack resistance: It considers if a trust management scheme is resistant to malicious attacks, such as malicious rating, self-promoting, and collusive behaviors.

2.1.2 Reputation management

While trust management plays an important role in guaranteeing meaningful services and interactions in cloud computing, reputation provides the basis of evaluating and quantifying trust levels. Habib identified a set of important parameters required to support reputation evaluation, including system performance (e.g., latency, bandwidth, availability, reliability, etc.), and security measures (e.g., physical security support, network security support, key management, etc.) [24]. Bradai proposed a reputation-based architecture to help users choose trustworthy peers in the cloud environment [25]. This architecture consists of three models, in which the first model evaluates the reputation given by the users, and the other two refine the reputation values to detect malicious ratings. Koneru proposed a reputation management system based on user recommendations [26]. Muralidharan proposed a novel reputation management system for volunteer clouds, in which computer owners can donate their computing resources [27]. This system assesses each participant's reputation according to the performance of service time, crash time, and correctness of service results. Although the three factors are important for service and reputation evaluation, the network status can easily affect different aspects of the performance. Therefore, it decreases the objectivity of a participant's reputation. Zhu proposed an authenticated reputation and trust management system for cloud and sensor networks [28]. The trust and reputation are calculated according to the performance and feedback of data processing, data privacy and data transmission. However, the system still depends on predefined service level agreement (SLA) and privacy level agreement (PLA).

Besides different factors chosen for reputation evaluation and management in various application contexts, unfair ratings and malicious attacks are also important issues which affect the performance and efficiency of a reputation management system. Yan proposed a centralized reputation management system for data access control in cloud computing [29]. It employs a Reputation Center (RC) as a trusted third party to manage and verify users' reputations. Moreover, it applies a punishment agreement to encourage good performances and honest voting. Wu proposed a reputation revision method which applies a novel filter to recognize the unfair ratings [30]. The reputation evaluation is based on the QoS, including response time, cost, and reliability, as well as users' ratings for prior service experience. The method applies similarity theory to distinguish abnormal evaluations, and calculates average ratings. Wang proposed an accurate and multi-faceted reputation scheme which detects

unfair ratings to improve the accuracy of reputation calculation [31]. The scheme firstly identifies suspicious users whose ratings deviate from others significantly, and then identifies collusive users through similarity clustering. After removing those malicious users and unfair ratings, it calculates the overall reputation value.

2.2 Role-Based Access Control in cloud computing

Role-Based Access Control (RBAC) was firstly proposed by Ferraiolo and Kuhn in [32]. The basic principle of RBAC is separating users and permissions by inserting different roles. Users are assigned to some roles based on their job functions and responsibilities. Each role has their corresponding operational permissions, and users can only obtain the permissions after activating their roles. RBAC simplifies the permission assignment and authorization management by grouping permissions according to the roles, as well as separating job duties. Moreover, by relating permissions only to roles instead of users, it is more scalable and easier to manage permissions in the cloud computing environment where a large amount of users are presented, and it is difficult to track each of their identities. However, RBAC manages permissions statistically according to predefined roles, without considering some dynamic aspects such as time, and service context. Bertino proposed a Temporal Role-Based Access Control (TRBAC) to address the periodic permission problems by enabling roles at different time intervals [33]. Joshi proposed a general temporal RBAC (GTRBAC) which allows a wider range of temporal constraints of role assignments and permission assignments by clearly defining a protocol of constraints, including active duration constrains, maximum number of activations of a role, and enabling/disabling roles [34]. Besides permission time intervals, dynamic service contexts are also considered to improve RBAC. Yu proposed a role and task based access control model (RTABC) to refine the permission assignment, by adding a task layer between permission sets and roles [35]. A task is a minimum function unit of an operation, and an operation process can contain a group of tasks. In RTABC, operating permissions are not directly assigned to roles, but tasks. Users can obtain permissions of certain tasks by their allocated roles, and the permissions are constrained by task state, task weight and time. Barati designed a new semantic role-based access control model for cloud computing [36]. The model enables recommendations of tasks which a user can possibly require the permissions of.

In RBAC [32], there are role hierarchies where senior roles can have permissions of their junior roles. However, sometimes it is necessary for a junior role to have a permission in order to perform a senior role's operations. Tang proposed a new RBAC model for cloud computing, in which there are two additional roles namely User Role (UR) and Owner Role (OR) [37]. Users can get permissions from owners to access some resources in the cloud. Na also proposed a role delegation RBAC which allows junior roles to be granted of their senior roles' permissions [38]. Delegation server and delegation protocols are employed in this method, in which a delegation server is responsible for the delegation, while the delegation protocol describes the delegation process. Lin proposed a scheme for cross-domain access control system in cloud computing, which integrates the RBAC with trust management. It establishes a set of associations with regard of roles between the local domain and other domains [39]. In order to perform various permission constraints and role delegations, Gitanjali proposed several policy specifications for RBAC [40]. The policies consist of permission delegation, role delegation, and also delegation of the access rights to cloud service providers. Moreover, Gitanjali designed the backup and restoration policies in case the service crashes and the data is lost.

2.3 Attribute-Based Encryption for cloud computing

Attribute-Based Encryption (ABE) has been recently well studied for data access control in cloud computing, because it does not require either the identities of the data requesters or their public keys, but only the attributes the requesters own [41]. Sahai and Waters proposed the initial ABE which evolved from Identity-Based Encryption [42]. In the initial ABE, an identity consists of a set of descriptive attributes. A user can use a private key for an identity ω to decrypt a ciphertext encrypted with an identity ω', if and only if ω match ω' over a certain threshold that makes it error-tolerant. The initial ABE was further extended to two varieties, Key Policy-Attribute Based Encryption (KP-ABE) and Ciphertext Policy-Attribute Based Encryption (CP-ABE). Key Policy-Attribute Based Encryption (KP-ABE) was proposed by Goyal in 2006 [43]. It is a public key cryptographic scheme built upon bilinear map and Linear Secret-Sharing Scheme (LSSS) [44]. In KP-ABE, a ciphertext is associated with a set of attributes during the encryption, while a secret key is generated based on an access policy, which is an access structure over a set of data attributes. To decrypt the ciphertext, the data attributes must satisfy the access structure. Chase proposed a multi-authority attribute based encryption scheme, which allows multiple authorities to control the

data access at the same time [45]. Each of the authorities maintains a set of attributes, and a user can only decrypt if the number of attributes he/she possesses is beyond the threshold of each authority. Wang proposed a KP-ABE scheme with constant ciphertext size no matter how many attributes are embedded [46]. Yan designed a secure personal health record system using KP-ABE in a cloud computing environment [47]. Although KP-ABE is believed secure for data access control in cloud computing, it comes with high computational cost, especially if there is a large number of attributes. Lv proposed an efficient and secure KP-ABE scheme for mobile cloud storage by outsourcing the KP-ABE key generation and decryption process to a trusted attribute authority [48]. Yu combined the KP-ABE and Proxy Re-Encryption (PRE) to achieve fine-grained data access control [49]. Moreover, the scheme also allows data owners to delegate most of the computation tasks to untrusted cloud servers without disclosing the underlying data contents.

The Ciphertext Policy-Attribute Based Encryption (CP-ABE) was proposed after KP-ABE, by Bethencourt [50]. Different from KP-ABE, a ciphertext in CP-ABE is built upon an access policy, while a private key is associated with a set of attributes. A user can decrypt the data only if its attributes embedded in the private key match the access policy. Lewko proposed a decentralized CP-ABE scheme which allows multiple authorities to issue access right and private keys with a part of the attribute set [51]. It solved the user collusion problem by applying user's unique global identity. Horvath further extended the decentralized CP-ABE scheme to reduce the computational burden of user revocation by removing the computations of revocation at cloud service providers and distributing them over service users [52]. Xu proposed a fine-grained document sharing system based on CP-ABE, which refines users into different classes according to their privileges to access files [53]. A document is divided into multiple segments and encrypted by hierarchical keys. A user with a higher security class key can derive all its lower level keys. Although Xu's scheme achieves fine-grained data access control, it is very difficult to manage both CP-ABE keys and hierarchical keys, especially where there are a large number of users in cloud environment. Wan proposed a hierarchical attribute-based solution in the cloud computing by organizing user attributes into different sets, where the same attribute can be assigned different values [54]. Moreover, it applied expiration time as an access constraint for user revocation, which made the scheme more efficient.

2.4 Security risks and privacy preservation in cloud computing

While embracing the dynamic service provisioning, ubiquitous data access, optimized resource allocation and low-cost infrastructure, cloud users are also faced with security problems of data confidentiality and privacy leakage.

2.4.1 Risks of data security in cloud computing

Outsourced data security

Apart from the security of data stored at cloud service providers, cloud computing brings additional security risks when the data owners outsourcing their sensitive data for data sharing on cloud servers. Although several works [55][56][57][58] were proposed to encrypt data before outsourcing to other users, and to use methods such as re-encryption and trusted third party to protect data confidentiality, they are facing problems in practical systems. On the one hand, they incur a large computational cost for encrypting all the data in the cloud, and lots of effort for user revocation and key management. On the other hand, it is difficult for data owners to know if the data has been tampered after outsourcing, especially when there are multiple owners of one file [24]. This fact may threat the integrity of data when sharing and outsourcing in the cloud environment.

Malicious rating

Comparing to encryption schemes with more computational cost, trust and reputation management provide a more computationally efficient way of protecting data security and access control in cloud environment. However, there could be malicious nodes or competitors who provide unfair ratings to affect the accuracy of trust and reputation. Although several works proposed to detect unfair ratings and malicious users [30][31][59], either of them requires a large amount of prior knowledge or use experience to assure the detection accuracy, or they can lead to false positive results to exclude eligible users.

Multi-tenancy security

Virtualization technology is now widely used to provide infrastructure sharing and resource optimization in cloud services. Cloud service providers can now support “Multi-Tenancy”, in which service users can store their data and deploy their data processing on the same physical hardware [60]. This service model poses security threats that a customer shares the

same physical hardware with its adversaries, and its resources (e.g., data, virtual machines, etc.) placed on the cloud servers can be attacked by the adversaries. Ristenpart shows how the malicious behaviors are carried out towards a customer when they share the same cloud server [60]. Factor proposed the Secure Logical Isolation for multi-tenancy cloud storage (SLIM) by predefining the principles to isolate the tenants' resources [61]. Yang integrated the RBAC and attribute check mechanism to determine which resource that a user can access [62]. Li also proposed a RBAC based scheme for multi-tenancy cloud service, by embedding predefined security duty separations [63]. Tang proposed a RBAC based control scheme and integrated trust management to set trust relations among tenants [64].

2.4.2 Privacy preservation in cloud computing

Privacy has been an essential issue that influences user's adoption of cloud services. The privacy violation of some famous cloud services such as Facebook and iCloud, have leaded to increased concerns in terms of either personal or commercial information leakage. Many existing works [65][66][67][68][69][70][71] propose to solve the problems of privacy violation. The proposed schemes can be divided into two categories: non-cryptographic and cryptographic schemes. The main non-cryptographic scheme is data perturbation. Wang proposed a privacy preserving scheme based on data anonymity [65]. It hides a part of the user information and prevents deducting sensitive information from disclosed data. Pan proposed a retrievable data perturbation method for privacy preservation based on random noise perturbation [66]. It adds random noise to perturb data values, but remains the data covariance unchanged so that users can retrieve the original values. Haas proposed a privacy preserving system for electronic health records by applying a trusted third party to control the data access, and prevent service providers of data storage from accessing and disclosing data [67]. For cryptographic methods, Wang proposed a privacy preserving scheme based on a public key based homomorphic authenticator, and also applied a third party auditor [68]. He also proposed a secure and efficient method for ranked key word searching to provide privacy preservation for outsourced data [69]. This method enables relevance ranking instead of sending undifferentiated results, and develops a one-to-many order-preserving mapping technique to protect sensitive information. Leng proposed a privacy preserving scheme for personal health records based on predefined policies, and applied proxy re-encryption for flexible encryption and access right delegation [70]. Narayan proposed a privacy preserving Electronic Health Record (EHR) system based on attribute-based cryptography [71].

Although there are many studies on protecting user data and information stored in the cloud environment, there are few works on privacy preservation in usage data, such as behavioral information, recently visited information, user devices and other usage histories which can be tracked and studied to violate user's privacy [72].

Chapter 3 Technical Preliminary and Access Control Schemes

This section introduces the technical preliminaries, and three data access control schemes that are implemented and evaluated in the thesis work. These schemes enable secure data access control in cloud computing based on trust and reputation.

3.1 Preliminary and cryptographic basics

PRE and ABE are two important cryptographic techniques which are highly related to our work. In order to best understand the PRE and ABE, we will briefly introduce them below. We firstly introduce the bilinear map that is the basic of ABE. Notably, knowledge of basic mathematics, such as finite fields, groups and elliptic curves, are required to understand the cryptographic algorithms. However, an extensive introduction of these areas is beyond the scope of this thesis.

3.1.1 Bilinear map

Bilinear map, or bilinear pairing, is a basis of many cryptography paradigms. There are different definitions of bilinear map, depending on the type of group and elliptic curve. Generally speaking, a bilinear map is an operation that combines elements of two groups to yield an element of a third group. Here we outline two commonly used definitions of bilinear map that are proposed by Boneh, Franklin and Lynn [73][74][75].

Symmetric pairing:

Definition 3.1: Let $\mathbb{G}$, $\mathbb{G}_T$ be cyclic groups of large prime order p, and $\mathbb{Z}_p$ be a ring of integers modulo p. Let g be a generator of $\mathbb{G}$. A bilinear pairing or bilinear map e is an efficiently computable function:

$$e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$$

such that

(i) Non-degeneracy: $e(g, g) \neq 1$. The map does not send all pairs in $\mathbb{G} \times \mathbb{G}$ to the element in $\mathbb{G}_T$. If g is a generator of $\mathbb{G}$, then $e(g, g)$ is a generator of $\mathbb{G}_T$.

(ii) Bilinearity: $e(g^a, g^b) = e(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p$.

The symmetric bilinear map is the original and simplest abstract definition of the pairing, and it is completely defined by the value it takes at $e(g, g)$. The Diffie-Hellman problem [9] can be solved in the bilinear map, since given g, g^x, g^y, g^z, by Bilinearity and nondegenracy $z = xy$ if and only if $e(g, g^z) = e(g^x, g^y)$. However, symmetric pairings can only be instantiated by using suitable super-singular elliptic curves.

Asymmetric pairing:

In order to allow a wider range of curves to be used, the asymmetric pairing looses the definition of symmetric pairing.

Definition 3.2: Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be cyclic groups of large prime order p. Assume the Diffie-Hellman problem is hard in $\mathbb{G}_1$. Let $\emptyset: \mathbb{G}_2 \to \mathbb{G}_1$ be an efficiently computable group isomorphism. Let g_2 be a generator of $\mathbb{G}_2$. Set $g_1 = \emptyset(g_2)$ (g_1 is the generator of $\mathbb{G}_1$). A bilinear pairing e is an efficiently computable function:

$$e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

such that

(i) Non-degeneracy: $e(g_1, g_2) \neq 1$.

(ii) Bilinearity: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{Z}_p$.

This modified definition allows a greater variety of pairings to be used on ordinary curves, and security proofs require only minimal changes because of the map $\emptyset$. However, there is a problem with hashing in this definition. It turns out that there is no known method to hash to an element of $\mathbb{G}_2$ such that its discrete *log* to some fixed base is unknown. This issue can complicate the design of some cryptosystems, and make the system designers give up asymmetric pairing in some cases.

3.1.2 Proxy Re-Encryption

Proxy cryptography was proposed by Blaze, Bleumer and Strauss [76], in which a proxy can convert ciphertexts from Alice into ciphertexts for Bob without seeing the underlying plaintext. In their Elgamal-based scheme, a proxy is entrusted to hold a re-encryption key b/a, which is created by Alice's secret key a and Bob's public key b. The proxy can use this re-encryption key to divert ciphertexts from Alice to Bob via computing the re-encryption

function. However, this scheme contains an inherent restriction: it is bidirectional; that is, the re-encryption key b/a can be used to divert ciphertexts from Alice to Bob and vice versa by computing $(b/a)^{-1}$. Therefore, this scheme is only useful when the trust relationship between Alice and Bob is mutual. Moreover, users are believed not to collude with the proxy in case the proxy wants to recover the secret keys.

In order to solve the problems within the scheme in [76], Ateniese improved the PRE algorithm by loosing the restrictions of trust relationship between users, and decreasing the security harm of collusion between users and the proxy [77]. The improved algorithm can be described as:

Key generation: User A's key pair is $\mathrm{pk_a} = (\mathrm{Z}^{\mathrm{a_1}}, \mathrm{g}^{\mathrm{a_2}})$, $\mathrm{sk_a} = (\mathrm{a_1}, \mathrm{a_2})$, where $\mathrm{g} \in \mathbb{G}$ is a random generator and $\mathrm{Z} = \mathrm{e}(\mathrm{g}, \mathrm{g}) \in \mathbb{G}_\mathrm{T}$ is a system parameter. $\mathrm{a_1}, \mathrm{a_2} \in \mathbb{Z}_\mathrm{p}$.

Re-encryption key generation: User A delegates the access right to B by publishing the re-encryption key $\mathrm{rk_{A\to B}} = \mathrm{g}^{\mathrm{b_2 a_1}} \in \mathbb{G}$, computed from B's public key $\mathrm{g}^{\mathrm{b_2}}$.

First-level encryption: To encrypt a message $\mathrm{m} \in \mathbb{G}_\mathrm{T}$ under $\mathrm{pk_a}$ in a way that it can only be decrypted by the holder of $\mathrm{sk_a}$, the user outputs the ciphertext as $\mathrm{c_{a,1}} = (\mathrm{Z}^{\mathrm{a_1 k}}, \mathrm{mZ^k})$, where $\mathrm{k} \in \mathbb{Z}_\mathrm{p}$ is a random number. Through first-level encryption, the decryption right can only be retained by the holder of $\mathrm{sk_a}$, and cannot be delegated to others.

Second-level encryption: To encrypt a message $\mathrm{m} \in \mathbb{G}_\mathrm{T}$ under $\mathrm{pk_a}$ in a way that it can be decrypted by A and other users who are delegated the decryption right by A, output $\mathrm{c_{a,2}} = (\mathrm{g^k}, \mathrm{mZ}^{\mathrm{a_1 k}})$.

Re-encryption: A proxy can divert a second-level ciphertext from A into a first-level ciphertext for B with re-encryption key $\mathrm{rk_{A\to B}}$, by computing $\mathrm{e}(\mathrm{g^k}, \mathrm{g}^{\mathrm{a_1 b_2}}) = \mathrm{Z}^{\mathrm{b_2 a_1 k}}$ and publishing $\mathrm{c_{b,2}} = (\mathrm{Z}^{\mathrm{b_2 a_1 k}}, \mathrm{mZ}^{\mathrm{a_1 k}})$.

Decryption: To decrypt a first-level ciphertext $\mathrm{c_{a,1}}$ with the secret key $\mathrm{a_1} \in \mathrm{sk_a}$, compute $\mathrm{m} = \frac{\mathrm{mZ^k}}{(\mathrm{Z}^{\mathrm{a_1 k}})^{1/\mathrm{a_1}}}$. To decrypt a ciphertext that has been re-encrypted, compute $\mathrm{m} = \frac{\mathrm{mZ}^{\mathrm{a_1 k}}}{(\mathrm{Z}^{\mathrm{b_2 a_1 k}})^{1/\mathrm{b_2}}}$. To decrypt a second-level ciphertext, compute $\mathrm{m} = \frac{\mathrm{mZ}^{\mathrm{a_1 k}}}{\mathrm{e}(\mathrm{g^k}, \mathrm{g})^{\mathrm{a_1}}}$.

This scheme has some improvements by making the delegation unidirectional, which means if A delegates the access right to B, B can decrypt A's message after re-encryption but A cannot access B's ciphertexts. This is because the re-encryption key $rk_{A\to B}$ cannot be used to compute $rk_{B\to A}$ if A and the proxy do not know B's secret key b_1. Moreover, even if B has been delegated the access right and colludes with the proxy, they can only recover the weak secret g^{a_1}, so that they cannot decrypt A's first-level ciphertext [77]. In our implementation, we applied this PRE scheme.

3.1.3 Ciphertext Policy-Attribute Based Encryption

The Ciphertext Policy-Attribute Based Encryption (CP-ABE) was proposed by Bethencourt [11]. A ciphertext in CP-ABE is built upon an access tree $\mathcal{A}$, while a private key is associated with a set of attributes γ. Generally, the CP-ABE algorithm has four operations that are defined in [11] as follows:

Setup: Choose a symmetric bilinear map e: $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, in which $\mathbb{G}$ is a bilinear group of prime order p and g is a generator of $\mathbb{G}$. In addition, choose a security parameter κ that determines the size of the groups. A hash function H: $\{0,1\}^* \to \mathbb{G}$, is employed to map an attribute from a binary string to a random group element.

The system will choose two random exponents $\alpha, \beta \in \mathbb{Z}_p$, and use the parameters defined in the bilinear map and hash function, to publish the public key PK as :

$$PK = \mathbb{G}, g, h = g^{\beta}, f = g^{1/\beta}, e(g,g)^{\alpha}$$

and the master key MK is (β, g^{α}).

Encrypt: Since the message M is encrypted under an access tree, we firstly need to define an access tree $\mathcal{A}$. The access tree $\mathcal{A}$ is constructed by three functions: *parent(x)*, *att(x)*, and *index(x)* (*x* is a node in the tree).

parent(x): The parent node of *x*.
att(x): The attribute of node *x*.
index(x): The index number of node *x*.

Let k_x be a threshold value, and denote $\mathcal{A}_x(\gamma) = 1$ if a set of attributes γ satisfy the access tree $\mathcal{A}_x$. The algorithm encrypts the message as follows:

(i) Choose a polynomial q_x for each node x in the tree $\mathcal{A}$, starting from the root node r. Then set the degree d_x of the polynomial q_x as $d_x = k_x - 1$, for each node x in the tree.

(ii) For each non-root node x, set $\mathrm{q_x}(0) = \mathrm{q_{parent(x)}}(index(x))$ and choose d_x other points of the polynomial q_x randomly; for the root node r, set $\mathrm{q_r}(0) = s$ (s is a random number: $s \in \mathbb{Z}_\mathrm{p}$) and choose d_r other points randomly.

(iii) Let Y be the set of leaf nodes in $\mathcal{A}$. The ciphertext is computed and published as:

$$\mathrm{E} = \left(\mathcal{A}, \widetilde{\mathrm{E}} = \mathrm{Me(g,g)^{\alpha s}}, \mathrm{E} = \mathrm{h^s}, \forall_\mathrm{y} \in \mathrm{Y}: \mathrm{E_y} = \mathrm{g^{q_y(0)}}, \mathrm{E'_y} = \mathrm{H\big(att(y)\big)^{q_y(0)}}\right).$$

KeyGen: The secret key SK is generated upon a set of attributes γ. The algorithm firstly chooses a random number $r \in \mathbb{Z}_\mathrm{p}$, and random $r_i \in \mathbb{Z}_\mathrm{p}$ for each attribute $i \in \gamma$. The secret key SK is computed as:

$$\mathrm{SK} = \left(\mathrm{D} = \mathrm{g}^{(\alpha+\mathrm{r})/\beta}, \forall_\mathrm{i} \in \gamma: \mathrm{D_i} = \mathrm{g^r \cdot H(i)^{r_i}}, \mathrm{D'_i} = \mathrm{g^{r_i}} \right).$$

Decrypt: First, define a recursive function DecryptNode(E, SK, x), which needs a ciphertext E, a secret key SK and a node x from $\mathcal{A}$. Then the decryption algorithm is designed as follows:

(i) For a leaf node x, we let $i = \mathrm{att(x)}$ and the function is computed as follows:

$$\mathrm{DecryptNode(E, SK, x)} = \begin{cases} \dfrac{\mathrm{e(D_i, E_x)}}{\mathrm{e(D'_i, E'_x)}} = \dfrac{\mathrm{e(g^r \cdot H(i)^{r_i}, g^{q_x(0)})}}{\mathrm{e(g^{r_i}, H(i)^{q_x(0)})}} = \mathrm{e(g,g)^{rq_x(0)}}, & \text{if i} \in \gamma \\ \perp, \quad \text{otherwise} \end{cases}$$

(ii) For a non-leaf node x, it calls DecryptNode(E, SK, z) for all child nodes z of x and stores the output as $\mathrm{F_z}$. $\mathrm{S_x}$ is an arbitrary k_x sized set of child nodes z such that $\mathrm{F_z} \neq \perp$. The $\mathrm{F_x}$ function is computed as:

$$F_x = \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)}, \quad \text{where } i = \text{index}(z), \text{and } S'_x = \{\text{index}(z) : z \in S_x\}$$

$$= \prod_{z \in S_x} \left(e(g,g)^{r \cdot q_z(0)}\right)^{\Delta_{i,S'_x}(0)}$$

$$= \prod_{z \in S_x} \left(e(g,g)^{r \cdot q_{\text{parent}(z)}(\text{index}(z))}\right)^{\Delta_{i,S'_x}(0)}$$

$$= \prod_{z \in S_x} \left(e(g,g)^{r \cdot q_x(i)}\right)^{\Delta_{i,S'_x}(0)}$$

$$= e(g,g)^{r \cdot q_x(0)}$$

(iii) Apply the above function to the root node r. If the access tree is satisfied by the attribute set γ, then A = DecryptNode(E, SK, r) = $e(g,g)^{r \cdot q_r(0)} = e(g,g)^{r \cdot s}$. The ciphertext is decrypted as:

$$\frac{\widetilde{E}}{\left(e(E,D)/A\right)} = \frac{\widetilde{E}}{\left(e\left(h^s, g^{(\alpha+r)/\beta}\right) / e(g,g)^{r \cdot s}\right)} = \frac{Me(g,g)^{\alpha s}}{e(g,g)^{\alpha s}} = M$$

The author also proposed a delegation function of private keys, which is an optional function in order to subtract attributes from the original attribute set γ to create a new more strict attribute set $\tilde{\gamma}$: $\tilde{\gamma} \in \gamma$. Since the new attribute set $\tilde{\gamma} \in \gamma$, the generated new $\widetilde{SK}$ is a secret key for $\tilde{\gamma}$ and it is equivalent to one received directly from the authority (e.g., the encryptor).

We adopt the idea of CP-ABE to relate a ciphertext to an access policy while public key pair is based on attributes, but modify the equations to integrate individual trust to the data access control schemes and implementation. We will introduce the schemes in Section 3.4 and the implementation in Chapter 4.

3.2 Trust and reputation management in the data access control schemes

Trust and reputation are mentioned in many literature resources that we introduced in Chapter 2. In this section, we provide technical details of trust and reputation management proposed by Yan [5], which are applied in the three data access control schemes and later implementations.

3.2.1 Trust assessment

The trust assessment model is built based on the user behaviors in the mobile social network [5]. The behaviors can be classified into three categories:

- Mobile calls
- Messages
- Local instant interaction

The trust level between two persons then can be automatically assessed based on the characteristics (e.g. number of calls, weight, etc.) of the above three categories. Table II gives the notations that will be used in the formula.

The formula of trust level assessment is designed as follows:

$$TL(i,j) = f\{TL'(i,j) + pl(i,j) * [\omega 1 * \theta(N\_c(i,j) + N\_c(j,i)) + \omega 2 * \theta(N\_m(i,j) + N\_m(j,i)) + \omega 3 * \theta(N\_i(i,j) + N\_i(j,i))] - pu(i,j)\}$$

Besides taking the behaviors as part of the parameters, the formula also considers the previous trust level, priority and punishment factor. In addition, the trust level assessment can be linked to a certain context based on key word extraction, in which the data owner can set the data access policies according to the context.

Table II The notations used in the trust assessment formula

Symbols	Description	Remark
$\boldsymbol{N\_c(i,j)}$	The number of calls made by i to j	
$\boldsymbol{N\_m(i,j)}$	The number of messages sent by i to j	
$\boldsymbol{N\_i(i,j)}$	The number of interactions initiated by i to j	
$\boldsymbol{TL(i,j)}$	The trust level of j assessed by i; $TL'(i,j)$ is the old value of $TL(i,j)$	
$\boldsymbol{\theta(I)}$	The Rayleigh cumulative distribution function $\theta(I) = \left\{1 - \exp\left(\frac{-I^2}{2\sigma^2}\right)\right\}$ to model the impact of integer number I, where $\sigma > 0$, is a parameter that inversely controls how fast the number I impacts the increase of $\theta(I)$. Parameter σ can be set from 0 to theoretically ∞, to capture the characteristics of different scenarios. We use $\theta(I)$ to model the influence of the number of calls, messages and interactions on social relationships.	
$\boldsymbol{\omega 1}$	The weight parameter to show the importance of voice call	$\omega 1 + \omega 2 + \omega 3 = 1$
$\boldsymbol{\omega 2}$	The weight parameter to show the importance of message	
$\boldsymbol{\omega 3}$	The weight parameter to show the importance of instant interaction	
$\boldsymbol{pl(i,j)}$	The priority level of j in i's social networks	
$\boldsymbol{pu(i,j)}$	The punishment factor of j in i's view	
$\boldsymbol{f(x)}$	The Sigmoid function $f(x) = \frac{1}{1+e^x}$, which is used to normalize a value into (0, 1) in order to unify an evaluated trust level into an expected scope	

3.2.2 Reputation Generation

The reputation model is proposed in [5], and it is composed of two parts: the reputation Rf contributed by the user feedback; the reputation Rp contributed by performance monitoring and reporting. Table III gives the notations that will be used in the formulas.

Table III The notations used in the reputation generation

Symbols	Description
$\boldsymbol{Rf}$	reputation composed by user feedback
$\boldsymbol{Rp}$	reputation composed by performance monitoring and reporting
$\boldsymbol{\theta(I)}$	The Rayleigh cumulative distribution function $\theta(I) = \left\{1 - \exp\left(\frac{-I^2}{2\sigma^2}\right)\right\}$
$\boldsymbol{O}$	$O = \sum_{m} C(k,t) * e^{-\frac{\lvert t_e - t\rvert^2}{\tau}}$
$\boldsymbol{t}$	time
$\boldsymbol{t_e}$	reputation evaluation time
$\boldsymbol{k}$	user k
$\boldsymbol{\tau}$	parameter for controlling time decaying,
$\boldsymbol{K}$	the total number of votes
$\boldsymbol{V(k,t)}$	user k's voting at time t
$\boldsymbol{C(k,t)}$	user k's credibility of providing feedback at time t, $C(k,t) \in [0,1]$

The reputation generation formulas are designed as follows:

User feedback $\boldsymbol{Rf}$:

$$Rf = \frac{\theta(K)}{O}\sum_{k=1}^{K} V(k,t) * C(k,t) * e^{-\frac{\lvert t_e - t\rvert^2}{\tau}},$$

Performance monitoring and reporting Rp:

$$Rp = \frac{P}{P + N + o} \ (o = 1)$$

Final reputation aggregated by combining Rf and Rp:

$$R(t_e) = \frac{\theta(K_{Rp})}{\theta(K_{Rp}) + \theta(K_{Rf})} Rp(t_e) + \frac{\theta(K_{Rf})}{\theta(K_{Rp}) + \theta(K_{Rf})} Rf(t_e)$$

3.3 Scheme 1: Controlling cloud data access based on reputation

Scheme 1 was proposed by Yan, Li and Kantola [29]. It intends to control data access in a cloud environment by applying PRE and reputation management. There are four kinds of entities in the system model: *Data Owners, Cloud Service Providers (CSP), Reputation Center (RC),* and *Users*. The *RC* is fully trusted and employed for reputation management, as well as helping the data owners check if the users meet the access policies. *CSP* is responsible for data storage, controlling data access including data re-encryption and issuing access right to eligible users. *Data Owners* are the ones who own the right of data access and altering, and *Users* are the ones who request for data access.

The proposed scheme:

We adopt Proxy Re-Encryption (PRE) algorithm that enables RC to issue a re-encryption key and access right to eligible entities. The procedures in the scheme can be grouped into four phases: System setup, New data creation, Data access, and User revocation.

System setup: Each entity in the system matins a public key and private key pair under the public key infrastructure of PRE. The global parameters for key generation are shared within the system.

New data creation: A data owner encrypts its data using a symmetric key *DEK*, and then encrypts the symmetric key *DEK* using the RC's public key *pk_RC*. The data can be either encrypted in First-level encryption of PRE so that the data can only be accessed and decrypted by the data owner itself, or encrypted in Second-level encryption that allows delegating access rights to other users. Then the data owner stores the data along with the

encrypted *DEK* to the CSP, and specifies an access policy to RC. The plaintext of data and DEK is hidden from CSP, thus provides data protection and privacy preservation. The CSP in this scheme functions as a proxy, which indirectly delegates an access right to an entity without learning anything about the secret data.

Data access: A user *u* firstly sends an access request to the CSP. The CSP would forward the request to RC, who is responsible for evaluating the user's latest reputation and decides if the user meets the access policy. If the user is eligible, RC will generate a re-encryption key *rk_RC->u*. The CSP conducts the ciphertext re-encryption based on the *rk_RC->u* received from the RC, and send the re-encrypted data to the user. The user can decrypt the data using its own private key *sk_u* and obtained *DEK*.

User revocation: User revocation is handled by RC who retains a list of revoked users that are no longer eligible to access the data. And the CSP blocks the revoked user from accessing the data.

We adopt punishment rate in an insurance agreement between an entity and RC. The punishment rate is set based on an entity's reputation level, in case of data disclosure from the entity. The higher the reputation, the lower the punishment rate in an agreement. In this way, the entities in a system are encouraged to provide honest behaviors and better performances. In practical systems, the insurance business can be run by RCs to guarantee legal data access, and compensate data owners if there is any illegal data disclosure. CSPs in a system will pay annual fee to RC for issuing re-encryption keys and managing reputations.

Detailed steps of access procedure are illustrated in Fig.1.

Step 1: The data owner firstly encrypts the data using *DEK*, denoted as E(*DEK*, *data*). And it encrypts the *DEK* using RC's public key *pk_RC*, denoted as E(*pk_RC*, *DEK*). Then the data owner uploads the encrypted data to the CSP, and specifies the data access policy to RC.

Step 2: The user sends a data access request to CSP for ciphertext E(*pk_RC*, *DEK*), and encrypted data E(*DEK*, *data*).

Step 3: CSP firstly verifies the user's ID to check if it is valid in the system. If the user's ID is valid, the CSP forwards the data request to the RC. Otherwise, the request is rejected.

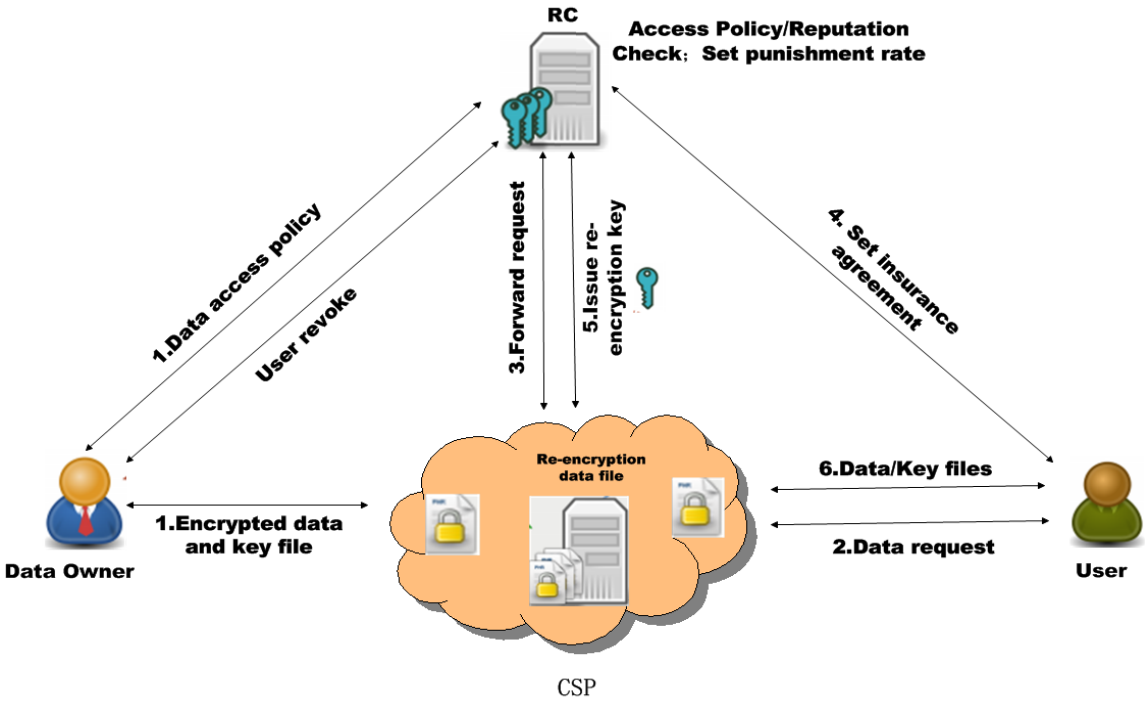

Figure 1 System model of cloud data access control based on reputation

Step 4: The RC evaluates the user's latest reputation, and decide if the user meets the access policy. If the user is eligible, the RC sets an insurance agreement based on the user's reputation level, in case of data disclosure. Otherwise, the request is rejected.

Step 5: The RC generates a re-encryption key $rk\_RC->u$=RG(sk_RC, pk_u), in which RG(sk_RC, pk_u) is the re-encryption key generation function, based on its own private key sk_RC and user's public key pk_u.

Step 6: The CSP conducts the ciphertext re-encryption R($rk\_RC->u$, E(pk_RC, DEK)) = E(pk_u, DEK), and sends the re-encrypted data E(pk_u, DEK) to the user. Therefore, the user can decrypt the data using its own secret key sk_u and obtained DEK.

If later on, the user wants to access the data again but no longer eligible, the RC will inform the CSP to block the user from accessing the data.

3.4 Scheme 2: Trust assessment controlled personal data access based on mobile social networking

Scheme 2 is designed for controlling cloud data access based on mobile social networking, by using trust assessment algorithm and CP-ABE for data encryption. There are three kinds

of entities in the system model: *Data Owner*, *CSP*, and *Users* who request the data. The *CSP* is considered semi-trusted, and is responsible for storing data and user validity check. The access right is fully controlled by *Data Owners*, who issue the decryption keys based on the *Users'* trust levels evaluated based on their social networking activities, behavior and experience.

The proposed scheme:

The scheme can be applied in an environment of mobile cloud computing. A mobile user can save its sensitive personal data at a data center offered by a CSP. For ensuring safe data access by other trustworthy users in the network, the mobile users make use of trust levels accumulated and analyzed from their individual mobile social networking records. The mobile user can issue secret keys to eligible users with sufficient trust to access the personal data at the CSP.

Trust level of a user is evaluated based on the activities, behaviors, and experiences in mobile networks. In the scheme, we divide individual trust into discrete levels. For example, *TL_i* represents user *i*'s trust level TL, and TL can be from 0 to $\bar{I}_{tl}$, where $\bar{I}_{tl}$ is the maximum level of TL.

In our scheme, plaintext is hidden from the CSPs in order to provide data protection and privacy preservation. The CSPs are responsible for data storage, verifying users' ID, and blocking eligible users from accessing the data. Trust evaluation and secret key issuing is handled by data owners themselves to ensure safe data access by trustworthy users. A user firstly sends an access request to a CSP, and the CSP will check if the user's ID in the system is valid. If it is the case, the CSP will forward the access request to the data owner, and the data owner will decide if the user who sends the request is eligible to access the data.

The user revocation is handled by the CSP based on the data owner's notifications about the non-eligible users. The encrypted data could be renewed by encrypting within a new access policy tree, or the access can be blocked by announcing the expired secret key. Although the CSP is semi-trusted, it is encouraged to perform well by applying the reputation mechanism to evaluate the CSP's performance. The CSP's reputation is based on the user feedback, and will be published to all users in the system.

Scheme algorithms:

The scheme contains two main algorithms: Trust assessment and CP-ABE based access control. Trust assessment algorithm is based on the formula in [5]. We adopt the idea of conventional CP-ABE, but integrate Individual Trust Level (TL) as an attribute in our modified CP-ABE algorithm. Concretely, the operations for our modified CP-ABE algorithm are: Setup, InitiateUser, CreateTrustPK, IssueTrustSK, Encrypt and Decrypt.

Setup: The Setup operation generates a system public key PK and a master key MK based on bilinear paring,

$$\mathrm{PK} = \{\mathbb{G}, \mathbb{G}_T, e, g, P, e(g,g)^y\},\ \mathrm{MK} = g^y$$

where P is a random point in $\mathbb{G}$, and $y \in \mathbb{Z}_p$.

InitiateUser: The InitiateUser takes public key PK and master key MK as inputs, and generates a user's public key PK_u and secret key SK_u,

$$\mathrm{PK\_u} = g^{mk_u},\ \mathrm{SK\_u} = \mathrm{MK} * P^{mk_u} = g^y * P^{mk_u}$$

where $mk_u \in \mathbb{Z}_p$. It also chooses a random hash function $H_{SK\_u}: \{0,1\}^* \rightarrow \mathbb{Z}_p$ from a finite family of hash functions.

CreateTrustPK: The CreateTrustPK operation generates CP-ABE public attribute key based on trust levels. It is executed by the data owners to encrypt the data and control the access right. The CreateTrustPK checks the TL related policies for data access, and output the public attribute key $PK\_(TL_i, u)$ for each attribute TL_i, where $i \in \left[0, \bar{I}_{tl}\right]$ and $\bar{I}_{tl}$ is the maximum level of TL. The $PK\_(TL_i, u)$ consists of two parts:

$$\mathrm{PK\_}(TL_i, u) = < \mathrm{PK\_}(TL_i, u)' = g^{H_{SK\_u}(TL_i)},\quad \mathrm{PK\_}(TL_i, u)'' = e(g,g)^{yH_{SK\_u}(TL_i)} >$$

Encrypt: The Encrypt operation takes the attribute public key $PK\_(TL_i, u)(i \in \left[0, \bar{I}_{tl}\right])$ to encrypt the symmetric key DEK, based on the access policy $\mathcal{A}$ related to TL. The output ciphertext CT is

$$\begin{aligned} CT_i = \langle\ & E_i = DEK \cdot (\mathrm{PK\_}(TL_i, u)'')^{R_i}, \\ & E_i' = P^{R_i}, \\ & E_i'' = (\mathrm{PK\_}(TL_i, u)')^{R_i} \rangle \end{aligned}$$

, where R_i is a random value and $R_i \in \mathbb{Z}_p$. The ciphertext CT consists of the tuple $CT = \langle CT_1, \cdots, CT_m \rangle$, and it iterates over all i=1,…,m ($m < \bar{I}_{tl}$), where m represents the number of selected TL in the access tree $\mathcal{A}$.

IssueTrustSK: The IssueTrustSK is executed after verifying user u' eligibility and if u' TL is equal or above the required TL level. It takes u′s pubic key as input and issues the TrustSK

$$SK\_(TL_i, u, u') = PK\_u'^{H_{SK\_u}(TL_i)} = g^{mk_u, H_{SK\_u}(TL_i)} \ (i = I)$$

Decrypt: The user decrypts the ciphertext to get the symmetric key DEK using its own secret key *SK_u'* and the TrustSK $SK\_(TL_i, u, u')$ issued by the data owner. The output DEK is

$$DEK = E_i \cdot \frac{e(E_i', SK\_(TL_i, u, u'))}{e(E_i'', SK\_u')}$$

Then the user can decrypt the encrypted data using the DEK.

Fig.2 shows the detailed procedures of data access control based on trust assessment and CP-ABE encryption in mobile social networks.

Step 1: The data owner conducts trust evaluations based on activities, behaviors and user experiences which are related to his/her usage in the mobile network. After the trust evaluation, the data owner determines the requirements of trust levels specified in his/her access policy. The data owner then encrypts the data using a secret symmetric key *DEK*, and encrypts the *DEK* using its own CP-ABE public key *PK_TL* which is based on specified trust levels. The encrypted data and symmetric key are denoted as E(*DEK*, data) and E(*PK_TL*, *DEK*). Then it uploads the encrypted data and DEK to the CSP, as well as sends the access policy based on the required trust level TL.

Step 2: A user firstly sends an access request to both the data owner and CSP.

Step 3: The CSP verifies the user's ID in the system in order to check if the user's ID is valid, or if the user is in the blacklist. If the user's ID is valid, the CSP forwards the access request to the data owner. Otherwise, the request is rejected.

Step 4: The data owner evaluates the user's trust level TL based on previous behaviors and activities. If the user is trustworthy, the data owner issues a secret key *SK_TL* based on the user's trust level TL, and also sends corresponding access policy $\mathcal{A}$. Otherwise, the request is rejected.

Step 5: After receiving the secret key *SK_TL* and access policy $\mathcal{A}$ from the data owner, the user again sends a data access request along with the access policy $\mathcal{A}$ to the CSP.

Step 6: The CSP checks if the access policy from the user is the same as that received from the data owner. If both of the policies match, the CSP sends the required data to the user, so that the user can decrypt using the *SK_TL*.

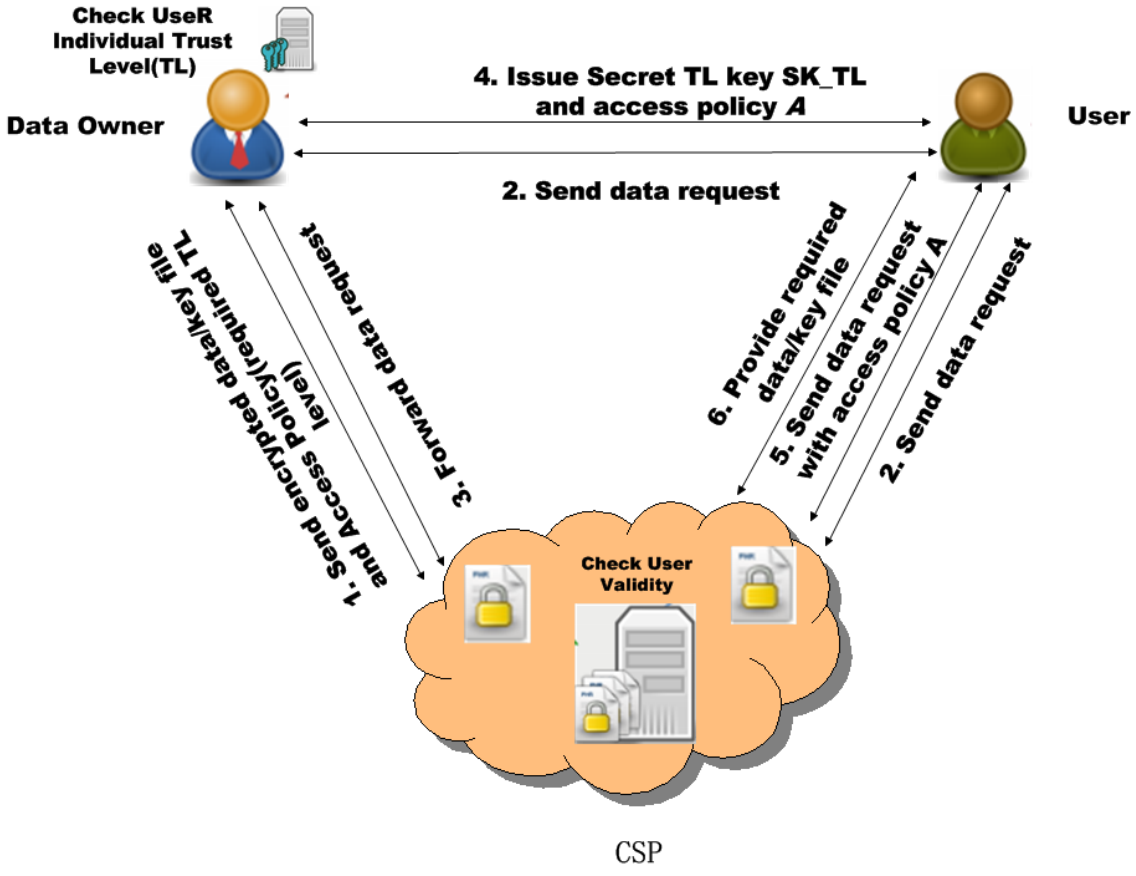

Figure 2 Procedure of cloud data access control based on trust assessment

3.5 Scheme 3: A scheme of heterogeneous data access control based on trust and reputation in cloud computing

Scheme 3 proposed a flexible multi-dimensional control on cloud data access [78]. It is a heterogeneous scheme which combines the techniques in Scheme 1 and Scheme 2. There are four kinds of entities in the system model: *Data Owners, Cloud Service Providers (CSP), Reputation Center (RC),* and *Users*. The *RCs* are fully trusted and employed for reputation management, as well as helping the data owners check if the users meet the access policies. The *CSPs* are responsible for data storage, controlling data access including data re-encryption and issuing access right to eligible users. *Data Owners* are the ones who own the right of access and altering, and *Users* are the ones who request for data access.

The proposed scheme:

In this scheme, we proposed multi-dimensional control on cloud data access based on individual trust evaluated by the data owners, and/or public reputation evaluated by one or multiple RCs. To be more concretely, a data owner firstly encrypts its data with a symmetric key *DEK*, and then the data owner can divide the *DEK* into several segments *K0, K1, K2… Kn*, K_{n+1}. *K0, K1, K2… Kn* are encrypted with public keys from different RCs which are employed to evaluate reputations and control data access. K_{n+1} can be encrypted with a public key *PK_TL* which is related to individual trust levels. After the data encryption, the data owner uploads the encrypted data and key segments to the CSP, and specifies the access policy to each of the RCs. In order to access data, a user needs to be authorized by all the RCs, and collect all key segments to recover the *DEK* for decryption.

The size of the key segments *K0, K1, K2… Kn* can be flexibly set by data owners, according to different application scenarios or security requirements. If a data owner would like to control data access only by itself, the symmetric key *DEK* will not be divided, and is encrypted with a public key *PK_TL* which is related to individual trust levels. If a data owner would like the RCs to the control data access, all the key segments are encrypted with the RCs' public keys.

User revocation is achieved by applying a blacklist which contains the ID of non-trusted or non-eligible users. The blacklist is managed by the CSP, and can be updated according to RC or data owners' notification and feedback.

Scheme algorithms:

The scheme consists of four main algorithms: Key generation, Symmetric key *DEK* division and combination, PRE, modified CP-ABE which is proposed in [23].

Key generation: Key Generation contains three kinds of keys: Symmetric key *DEK* for data encryption, public key pairs for PRE, and key pairs for CP-ABE. The key generation for CP-ABE consists of system public key *PK*, master key *MK*, user public key pairs and Individual Trust public key and secret key. The key generation can be conducted by users or by a trustworthy user agent.

Symmetric key *DEK* division and combination: Key division is operated by the data owner based on its data access control policy. The symmetric encryption key *DEK* is divided into n+1 parts, where n is the number of RCs which are employed by the data owner to control its data access based on the access policy. Key combination is operated by the user who receives all pieces of the symmetric key *DEK*, and aggregates all partial keys together to get a complete key *DEK* for decryption.

PRE: The data owner encrypts *n* pieces of partial symmetric key *DEK* using corresponding RC's PRE public key, and stores the encrypted data and key files in the CSPs. The RCs control data access right by evaluating the access policy and users' reputation, and conduct re-encryption key generation if a user is eligible for accessing the data. The CSPs conduct the re-encryption and send the re-encrypted data to the user.

CP-ABE: CP-ABE is applied for the purpose of integrating the individual trust level (TL) into the data access control mechanism, and controlling access right by the data owner itself. One piece of symmetric key *DEK* is encrypted using the data owner's Individual Trust public key, and is stored in the CSPs. After verifying the individual trust level of a user who requires the data, the data owner will then issue the user an Individual TL secret key, and inform the CSPs to send the encrypted data to the user.

Fig.3 illustrates the detailed procedures of two-dimensional data access control in the proposed scheme.

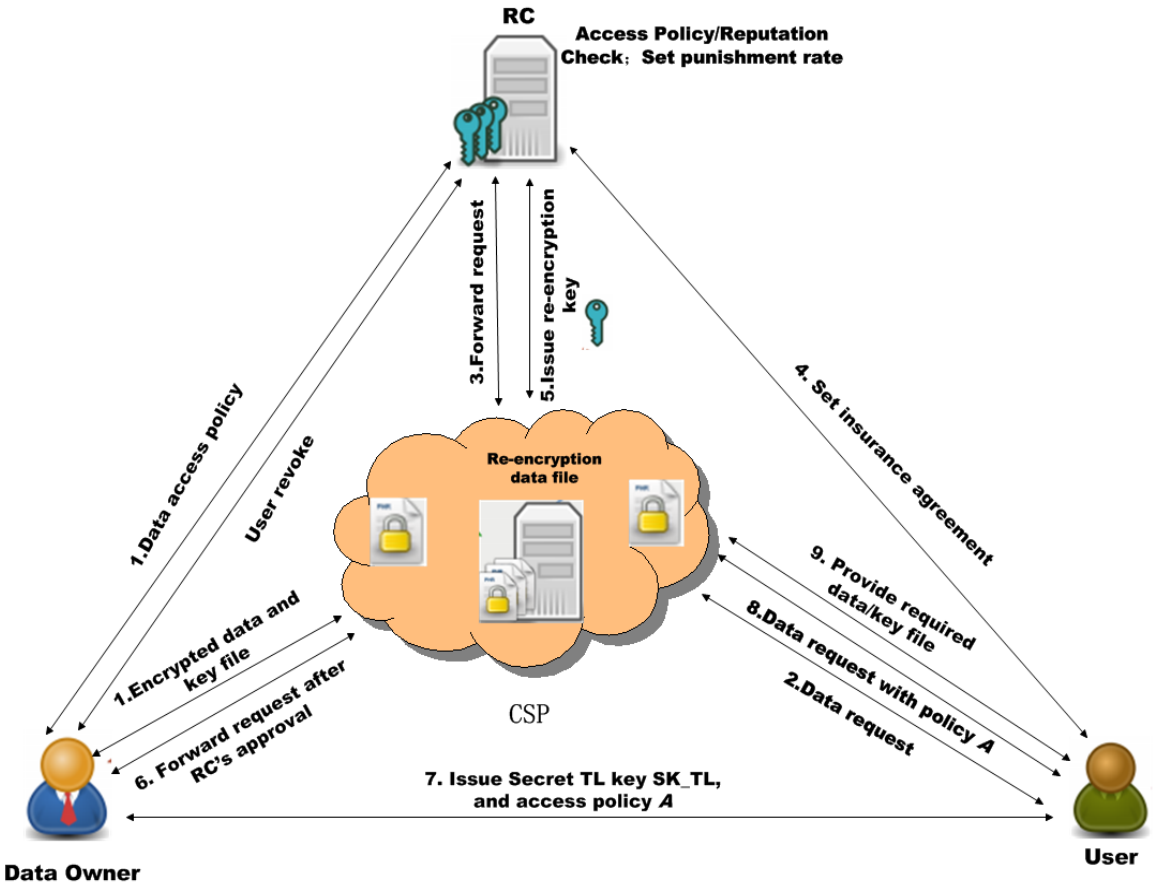

Figure 3 Procedure of cloud data access control based on heterogeneous scheme

Step 1: The data owner encrypts its data using a symmetric key *DEK*, and divides the *DEK* into two segments*:* K_0 and K_1. K_0 is encrypted with the RC's public key *pk_RC*, and K_1 is encrypted with a public key *PK_TL* which is related to individual trust levels. The encrypted data is denoted as E(*DEK*, data), and the key segments are denoted as E(*pk_RC*, K_0) and E(*PK_TL*, K_1). Then the data owner uploads the encrypted data to the CSP, and specifies an access policy to both the CSP and RC.

Step 2: The user sends an access request to the CSP, and waits for responses.

Step 3: The CSP verifies the user's ID and checks the blacklist in order to decide whether to forward the access request to the RC. If the user's ID is valid and it is not in the blacklist, the CSP will forward the request to the RC. Otherwise, the request is rejected.

Step 4: The RC evaluates the user's reputation, and decides if the user meets the access policy. If the user is eligible, the RC will set an insurance agreement with the user in case of illegal data disclosure. Otherwise, the request is rejected.

Step 5: The RC issues the re-encryption key *rk_RC->u*=RG(*sk_RC*, *pk_u*), in which RG(*sk_RC*, *pk_u*) is the re-encryption key generation function, based on the RC's own private key *sk_RC* and user's public key *pk_u.*

Step 6: After receiving the re-encryption key *rk_RC* from the RC, the CSP forwards the access request to the data owner.

Step 7: The data owner evaluates the user's trust level TL based on previous behaviors and activities. If the user is trustworthy, the data owner issues a secret key *SK_TL* based on the user's trust level TL, and also sends corresponding access policy $\mathcal{A}$. Otherwise, the request is rejected.

Step 8: After receiving the secret key *SK_TL* and access policy $\mathcal{A}$ from the data owner, the user again sends a data access request along with the access policy $\mathcal{A}$ to the CSP.

Step 9: The CSP checks if the access policy from the user is the same as that received from the data owner. If both of the policies match, the CSP conducts the ciphertext re-encryption R(*rk_RC->u*, E(*pk_RC*, K_0)) = E(*pk_u*, K_0), and sends E(*DEK*, data), re-encrypted data E(*pk_u*, K_0) and E(*PK_TL*, K_1) to the user.

Chapter 4 Scheme Implementation

In this chapter, we will firstly introduce the implementation design of each scheme. Then we will present the implementation of the essential algorithms, and SSL-based network communications.

4.1 Implementation design

We evaluate the three schemes by implementing their basic algorithms and functions. The implementation of Scheme 1 contains four main function blocks, shown in Fig.4: Encryption, Decryption, PRE Algorithm, and Reputation Evaluation. The implementation of Reputation Evaluation is based on the equations described in Section 3.2.2.

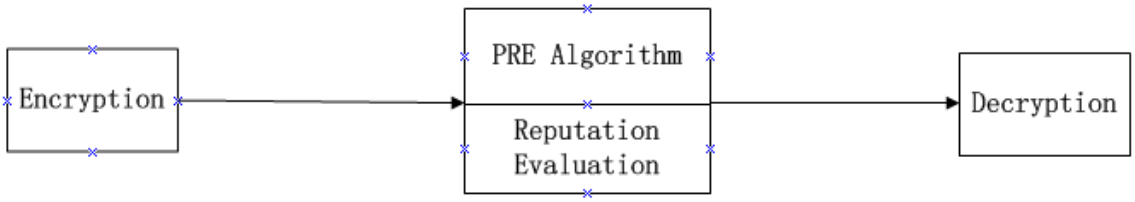

Figure 4 Main function blocks: reputation based cloud data access control

Fig.5 shows the main function blocks in Scheme 2: Encryption, Decryption, Individual Trust Level (TL) Assessment, and TL based CP-ABE algorithm. The TL based CP-ABE includes Master Key Generation, Public/Secret Key Generation, TL_Public/Secret Key Generation, TL_Encryption/Decryption. The implementation of Individual TL Generation and Assessment is based on the equations presented in Section 3.2.1.

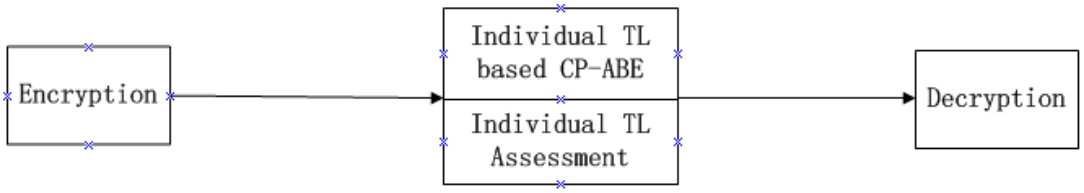

Figure 5 Main function blocks: Individual TL based CP-ABE scheme

Except for cryptographic algorithms and reputation/TL assessment, Scheme 3 has additional blocks: Key Division/Combination, in order to flexibly control data access by either the data owner, or RCs or both, shown in Fig.6.

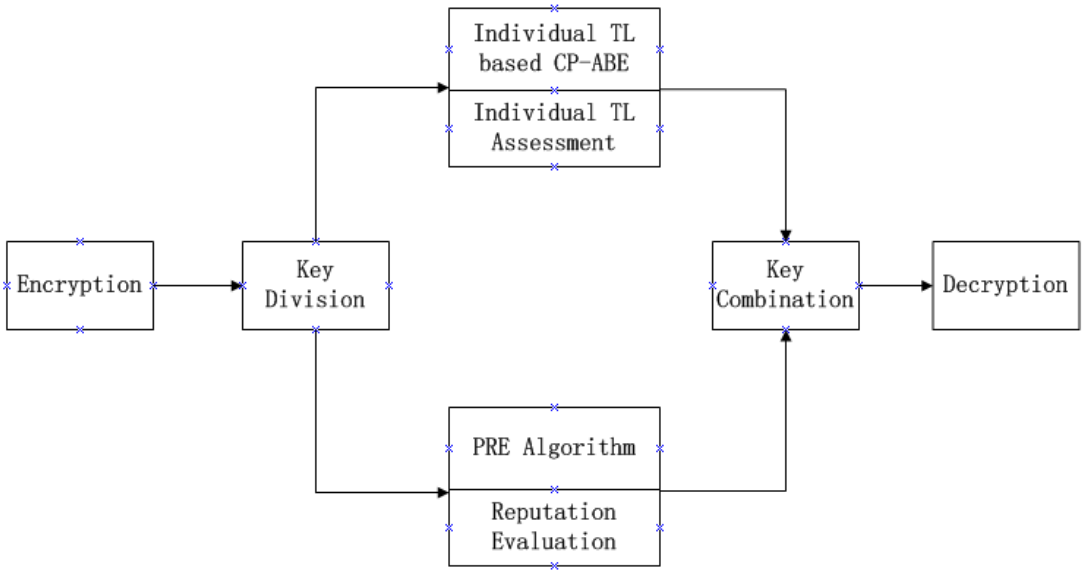

Figure 6 Main function blocks: heterogeneous scheme

The communication between each entity is based on client/server mode, through different ports on localhost. And the data transmission is secured by applying the SSL protocol. Since we focus on the performance of data access control schemes in this thesis, the collection of network data, such as voting and user behavior is beyond our scope. These data are simulated to support the scheme evaluation.

4.2 Basic functions

In the following sections, we introduce the implementation of main functions related to the three schemes.

4.2.1 Implementation of PRE

PRE belongs to Asymmetric Cryptography, and the algorithm is based on Elliptic Curves and Pairing. In order to implement PRE, we utilized Multi-precision Integer and Rational Arithmetic C Library (MIRACL, http://www.certivox.com/miracl), and JHU-MIT Proxy Re-encryption Libarary (http://spar.isi.jhu.edu/~mgreen/prl/).

Library overview

MIRACL is a cryptographic C library that supports multiple crypto algorithms, such as AES, RSA, DSA digital signature and so on. It is applied in our implementation because MIRACL is widely used for Elliptic Curve Cryptography, and is particularly adept at Pairing-Based Cryptography. MIRACL contains efficient algorithms for computations with very large numbers, and it has inline C++ wrapper, which can greatly simplify program development in C++. MIRACL has a new data type defined as *big* for large integers, and some data structure, such as *ZZn* and *ECn*, which are used in the program. These data types are defined to simplify the manipulation of points and computation on elliptic curves. MIRACL also defines many routines to support the computation of the defined data types, such as *add*, *divide*, and *pow*. Functions like *big_to_bytes* and *bytes_to_big* are defined to support the transition between large rational numbers and binaries.

JHU-MIT Proxy Re-encryption Library is a C++ library designed for proxy re-encryption. It provides many utility functions, such as parameter generation, serialization, and data type transition. The JHU-MIT library is based on MIRACL.

Essential algorithms:

Global parameter generation: One important difference between proxy re-encryption and other public cryptography methods is that all entities in a system would share the same global parameters. The global parameters define the size of prime field, and initiate elliptic curve computation. *generate_params(global_parameters)* function firstly asks for a seed to initiate the random number generation using *irand(seed)*. Then it defines the order of group $\mathbb{G}$, which is 160 bits, and generates the size of the prime field P, which is 512 bits. Finally it generates the generator of group $\mathbb{G}$, and the value of $\mathbb{Z}$ which is a basic value of pairing. All the global parameters are written into the *publicParam* file and shared in the system.

Public/Secret key generation: Every entity in the system uses the global parameters to generate their own public/secret key pairs. *keygen(params, publicKey, secretKey)* function takes global parameters as input, and outputs a public key and a secret key. The secret key is kept to itself, and the public key is used by others for encryption or key delegation. The *secretKey* contains two *Big* numbers *a1* and *a2*, generated by *rand()*. The *publicKey* consists of g^{a_2}, and Z^{a_1} which are generated using *ecap(global_parameters)* by doing the distortion

map and fast pairing. The public key size is 1024 bits, which is two times of the field size of the underlying elliptic curve.

Key delegation: One entity can generate a re-encryption key to delegate the access right to other users. The key delegation function *PRE_delegate(global_parameters, SK, PK, resKey)* takes global parameters, its own secret key *SK*, other user's public key PK as input, and outputs the re-encryption key *resKey*. The re-encryption key is computed as $resKey = PKSK$.

Ciphertext: The operations of ciphertext contain Encryption, Decryption, and Re-encryption.

PRE_level2_encryption(global_parameters, plaintext, PK, ciphertext) takes global parameters, a public key PK, and plaintext as input, and outputs the ciphertext. The function firstly transforms the plain data from text to *m*, which is a *Big* data type, and randomly generate value *k* for the latter computation. The first part of the ciphertext *c1* is computed as *k*g* (g is the generator of group $\mathbb{G}$), and the second part *c2* is computed as *m*pow(PK, k)*, where *pow()* is the power operation defined in MIRACL.

PRE _decryption() takes global parameters, ciphertext, its own secret key *SK* as input, and outputs the plaintext. The plaintext is computed as *c2/pow(c1, inverse(SK))*.

PRE_reencryption(global_parameters, originalCiphertext, rekey, newCiphertext) takes global parameters, delegation key *resKey*, original ciphertext as input, and outputs new ciphertext. The function remains the second part *c2* of the original ciphertext, and computes the first part c1 as *ecap(c1, resKey, params.q, params.cube, res1)*, where *params.q, params.cube* belongs to global parameters and *res1* is the output.

Utility functions: There are some utility functions defined to support the PRE algorithm. Table IV presents the main utility functions in the implementation.

Table IV Utility functions

Functions	Description
ReadFromPlaintext()	Read plaintext from file
ReadFromParam()	Read parameters from file
ReadPublicKeyFile()	Read public key from file
ReadSecretKeyFile()	Read secret key from file
PRE_PK_serialize()	Serialize public key to binaries for transmission
PRE_SK_serialize()	Serialize secret key to binaries for transmission
PRE_PK_deserialize()	Deserialize public key to PRE_PK data type for computation
PRE_SK_deserialize()	Deserialize secret key to PRE_PK data type for computation
PRECiphertext_serialize()	Serialize ciphertext to binaries for transmission
PRECiphertext_deserialize()	Deserialize ciphertext to ciphertext data type for computation

4.2.2 Implementation of CP-ABE

The TL-based CP-ABE proposed in Scheme 2 is a class of pairing based cryptography. It applies Individual TL as the attribute in the access policy tree. Public/Secret key pair is related to the attributes, while the ciphertext is related to an access policy. In order to implement the pairing based CP-ABE algorithm, we applied the Pairing-Based Cryptography (PBC) Library (http://crypto.stanford.edu/pbc/), which is an open C library that supports multiple cryptographic methods and performs the mathematical operations underlying

pairing-based cryptosystems. Bison/Yacc parser generator was applied in the implementation for policy and attribute parsing.

Library overview

PBC Library is an open C library that is widely used in many cryptosystems, especially in pairing-based cryptosystems. It is designed to improve the speed and portability of the implementation, and it provides routines such as elliptic curve arithmetic and pairing computation. One important merit of PBC is that it provides hands on functions that can be used by programmers who can concentrate on the properties of the cryptosystems, rather than the underneath number theories. For managing data utilities like bit/byte operation or dynamic arrays, we apply GLIB, which is a low-level system lib written in C. GLIB provides advanced data structures and utility functions.

Bison/Yacc (http://www.gnu.org/software/bison/) is applied as a parser generator to convert readable text into computer languages. By describing the input structure (called grammar rules), designing code to be invoked when these rules are recognized, and functions to deal with the basic input, the user can use Yacc to parse the input stream into computer languages and invoke the related operations. Bison is a general parse generator and upward compatible with Yacc.

Essential algorithms

Policy/Attribute parser: We apply Bison/Yacc as a Policy/Attribute parser to convert input policy and attributes into computer language for cryptographic computation. After downloading and installing Bison, we designed the grammar rules for the parser and the functions to invoke and analyze the input streams according to the grammar rules. The grammar rules and parse functions are written in the file *policy.y*, where *.y* format means that the codes are written in Yacc format. Fig.7 shows the grammar rules for parsing the input policy/attribute.

```
result: policy { final_policy = $1 }

number:   INTLIT '#' INTLIT          { $$ = expint($1, $3); }
        | INTLIT                     { $$ = flexint($1);    }

policy:   TAG                        { $$ = leaf_policy($1);         }
        | policy OR  policy          { $$ = kof2_policy(1, $1, $3); }
        | policy AND policy          { $$ = kof2_policy(2, $1, $3); }
        | INTLIT OF '(' arg_list ')' { $$ = kof_policy($1, $4);      }
        | TAG '=' number             { $$ = eq_policy($3, $1);       }
        | TAG '<' number             { $$ = lt_policy($3, $1);       }
        | TAG '>' number             { $$ = gt_policy($3, $1);       }
        | TAG LEQ number             { $$ = le_policy($3, $1);       }
        | TAG GEQ number             { $$ = ge_policy($3, $1);       }
        | number '=' TAG             { $$ = eq_policy($1, $3);       }
        | number '<' TAG             { $$ = gt_policy($1, $3);       }
        | number '>' TAG             { $$ = lt_policy($1, $3);       }
        | number LEQ TAG             { $$ = ge_policy($1, $3);       }
        | number GEQ TAG             { $$ = le_policy($1, $3);       }
        | '(' policy ')'             { $$ = $2;                      }

arg_list: policy                     { $$ = g_ptr_array_new();
                                       g_ptr_array_add($$, $1); }
        | arg_list ',' policy        { $$ = $1;
                                       g_ptr_array_add($$, $3); }
;
```

Figure 7 Grammar rules for policy/attribute parse

In the above grammar rules, the left column is the designed format for the input policy, and the right column is the function which should be invoked when recognizing the input policy. In the policy format column, the tokens (the basic element in the expression, e.g. 'TAG', 'number', etc.) are split in space. And the expression contains three kinds of operation, which are AND/OR, arithmetic operation, and comparison. The comma ',' is defined to separate different policies in the policy array. In the right column, there are functions that are defined to analyze the input policies after their corresponding expressions are recognized by the parser. The dollar symbol "$$" is a pseudovariable which represents the returned value of the function.

After the design of policy/attribute parser, it is necessary to transform the *policy.y* file into *policy.c* file, in order to include it in the program.

```
$ bison   input-file   -o   output-file
```

The above bison command is used to transform *.y* file to *.c* file which can be invoked in the program. The input-file is the *.y* file, and the output-file is the specified output *.c* file. If the program is written in C++, the postfix of the input file could be assigned as *.ypp*, so that the output file is in *.cpp* format.

System initiation: The System initiation includes pairing type define, system Master Key (MK) generation, and system Public Key (PK) generation. In PBC Library, there are two basic data types that are widely used in our program.

element_t: The basic element of algebraic structure.

pairing_t: The pairings where elements belong to.

element_t is used to declare algebraic variables, and *pairing_t* is used to declare a pairing. We apply the symmetric pairing Type A in our implementation, and initiate the pairing buffer before applying the pairing operation. Table V lists the functions used to initiate the elements.

Table V List of functions for element initiation

Functions	Description
element_init_G1(element_t e, pairing_t pairing)	Initiate an element in group $\mathbb{G}$
element_init_GT(element_t e, pairing_t pairing)	Initiate an element in group $\mathbb{G}_T$
element_init_Zr(element_t e, pairing_t pairing)	Initiate an element in integer ring Z_r
*pairing_init_set_buf(pairing_t pairing, const char *s, size_t len)*	Pairing initiation, where **s* stands for the description of the pairing type, and *len* is the maximum of the buffer length.

After element initiation, we can apply *element_pow_zn(element_t x, element_t a, element_t n)* for $x = a^n$ computation, and *pairing_apply(element_t p, element_t h, element_t a, pairing_t pairing)* to conduct the pairing operation. Function *node_initiate(&pub_u, &prv_u)* is to generate an entity's public key pk_u and secret key sk_u. It takes the system public key PK and the master key MK as input, and outputs its public/secret key pairs.

Individual TL Public/Secret key pair generation: The Individual TL Public/Secret key pair integrates the input attributes in the generation process. And in this scheme, the attributes are different TLs. For TL public key *PK_TL*, it traverses all the Individual TL to

cover all the attributes in the encryption. The program is designed to firstly read the input attributes, and invoke the parser by calling the function *parse_attribute(GSList* alist, char** argv)*, where *argv* represents the input attributes and *alist* is a linked list for storing the parsed attributes. Then the program calls *pub_unserialize(GByteArray* pk, int)* to unserialize the system PK to a data structure for computation. Function *trust_pub_keygen(pub_t* pk, char** attrs)* takes the system public key pk and parsed attributes as input, and outputs the Individual TL public key *PK_TL*. Fig.8 shows the work flow of the *PK_TL* generation.

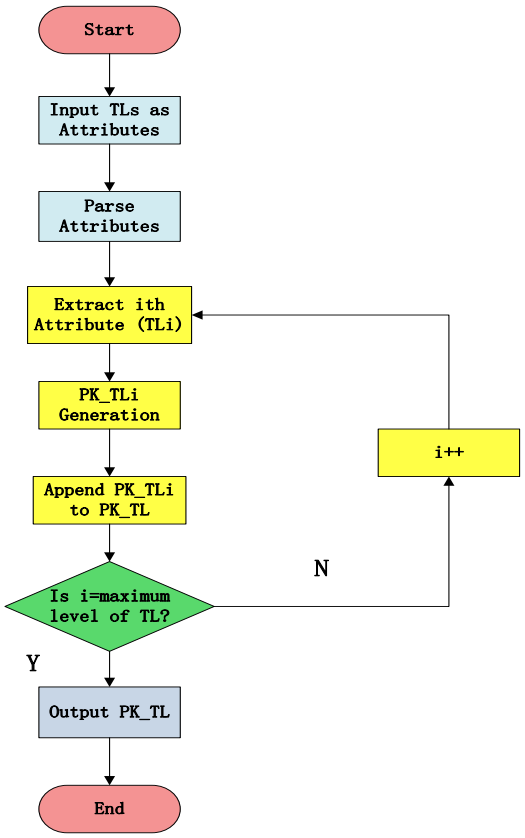

Figure 8 PK_TL generation

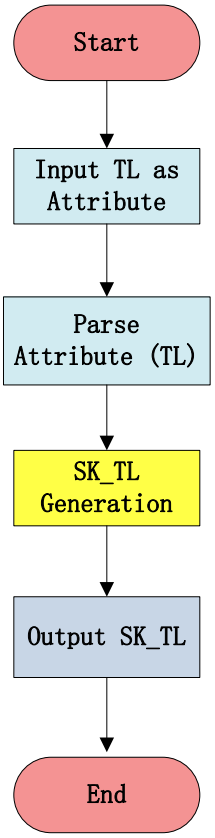

Figure 9 SK_TL generation

The generation of Individual TL secret key *SK_TL* only includes the issued TL as the attribute. Function *trust_prv_keygen(pub_u* pk_u, pub_t* pub, char** attr)* takes user's public key *pk_u*, system public key *pub*, and issued TL as input, and returns the TL secret key *SK_TL*. Fig.9 shows the work flow of *SK_TL* generation.

In order to invoke the function of TL key generation, we design the command syntax to assure that all the compulsory parameters are prepared.

Command syntax for *PK_TL* generation:

trust_pub_keygen [-h] [-o FILE] [INPUTFILE] [TL_1] [TL_2] [TL_3]……[TL_{max}]

Option:

-h	print command format message
-o	specify the name of the output file
INPUTFILE	the system public key file required for PK_TL generation

Command syntax for *SK_TL* generation:

trust_prv_keygen [-h] [-o FILE] [INPUTFILE1] [INPUTFILE2] [TL]

Option:

-h	print command format message
-o	specify the name of the output file
INPUTFILE1	the system public key file required for PK_TL generation
INPUTFILE2	the user's public key file required for PK_TL generation

Ciphertext-Policy Encryption/Decryption: The Encryption algorithm *trust_enc(pub_t* pk, trust_pub* pk_tl, element_t m, char* policy)* takes the system public key *pk*, the TL public key *pk_tl*, the plaintext, and the policy as input, and outputs the ciphertext which is encrypted according to the policy tree. It firstly parses the input policy by applying the function *parse_policy_postfix(char* policy)*, and then fills out the policy tree with required attribute nodes (Individual TLs) and threshold k. *fill_trust_policy(policy_t* p, pub_t* pub, trust_pub* pub_tl, element_t m)* is the essential function in the encryption algorithm. It implements a recursive algorithm which extracts every leaf node (Individual TLs with no child nodes) in the policy tree, and conducts encryption with the leaf node. In our scheme, the policy tree normally has two layers since we take Individual TL as the attribute. The bottom layer consists of all the attributes for Individual TLs. The second layer is the result of AND/OR

operations according to the specified access policy. Fig.10 shows the work flow of the encryption algorithm.

Decryption bswabe_trust_dec(cph_t cph, trust_prv* prv_tl, prv_u* prv_u, pub_t* pk)* takes ciphertext, TL secret key, user's secret key and system public key as input, and outputs the plaintext. It also implements a recursive algorithm *dec_trust_policy(policy_t* p, trust_prv* prv_tl, prv_u* prv_u, pub_t* pub, element_t m)* that invokes itself until it reaches all the leaf nodes to get the Individual TLs in the policy tree. By extracting all the leaf nodes, it compares each leaf node with the issued Individual TL recognized in the TL secret key until it finds out the match. Fig.11 presents the work flow of the decryption algorithm.

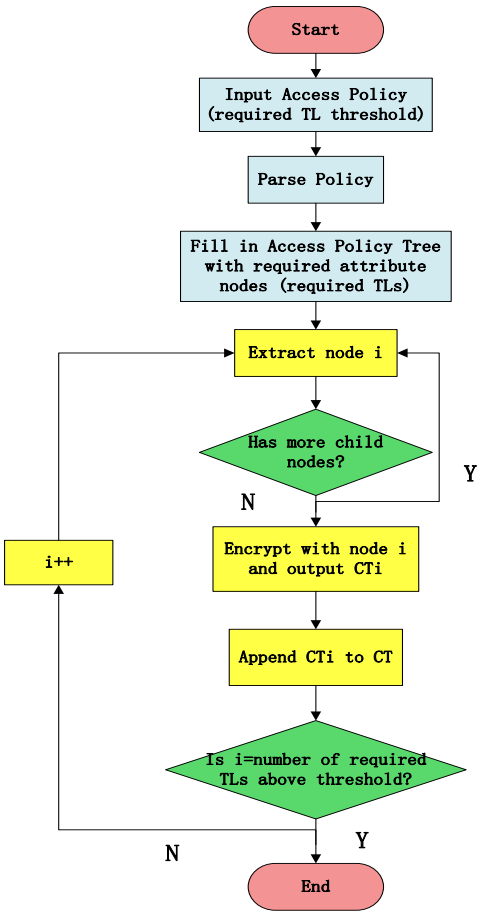

Figure 10 Encryption work flow

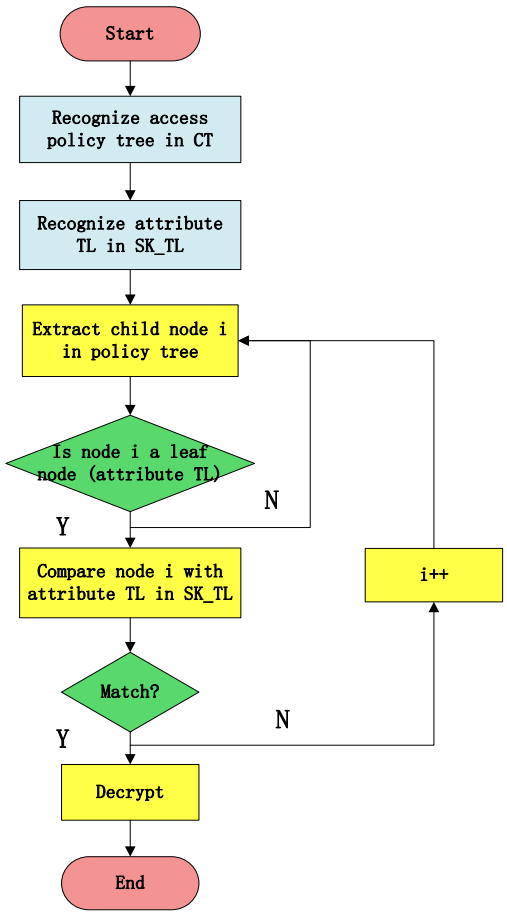

Figure 11 Decryption work flow

Command syntax for Encryption:

trust_enc [-h] [-o FILE] [INPUTFILE1] [INPUTFILE2] [POLICY]

Option:

-h	print command format message
-o	specify the name of the ciphertext file
INPUTFILE1	the system public key file required for encryption
INPUTFILE2	the PK_TL file required for encryption
POLICY	specify the access policy (e.g., TL > 3)

Command syntax for Decryption:

trust_dec [-h] [-o FILE] [INPUTFILE1] [INPUTFILE2] [INPUTFILE3]

Option:

-h	print command format message
-o	specify the name of the output file
INPUTFILE1	the system public key file required for decryption
INPUTFILE2	the user's secret key file required for decryption
INPUTFILE3	the SK_TL file required for decryption

4.2.3 Implementation of reputation/ individual TL assessment model

Reputation model:

The reputation model consists of two parts: User Feedback and Performance Monitoring. Since we focus on the reputation algorithm implementation and evaluation, the collection of network data, for example users' voting and monitoring data, is not in the scope of this implementation. The data used in this section is simulated according to the requirements of the algorithm.

We included the properties of user's reputation and punishment rate into an object of *Client*, and owner information and access requirements into an object of *DataFile*. Fig.12 and Fig.13 show the structure of the two objects.

```
class Client {
public:
    int userId;
    double punishmentRate;
    double userRepLevel;

    Client(){
        this->punishmentRate = punishmentRate;
        this->userRepLevel = userRepLevel;
    }

    ~Client(){
    }

    double userFeedback();
    double performanceMonitoring();
    double reputationLevel(double feedback, double perMonitoring);
    double renewClientRep(double oldCredits, double feedback, double finalReputation);
    void setPR();
};
```

Figure 12 Client object structure

In Fig.12, the *Client* object contains its user id, reputation level, and punishment rate as public variable. Additionally, *userFeedback()* is designed to give feedback to other users in the system, while *performanceMonitoring()* stores its network performance data. *reputationLevel()* and *renewClientRep()* is designed to evaluate and update user's reputation. In Fig.13, the *DataFile* object contains the file information and its access policy specified by the data owner. RC can retrieve the file information and access policy after receiving a request, and verify if the requester is eligible.

```
class DataFile {
public:
    map<int, accessNum> userAccessNum;
    map<string , fileInfo> datafileInfo;

    DataFile(){
    }

    ~DataFile(){
    }

    int getOwnerId(char* filename);
    fileInfo getFileInfo(string filename);
    bool addClient(int id, int maxAccessNum);
};
```

Figure 13 DataFile object structure

The algorithm is implemented according to the equations introduced in Chapter 3. Table VI lists the values defined for some parameters and variables in the equations.

Table VI Parameter values and variable range

Param/Variable	Value/Range
τ	1.0
σ	100.0
monitoring data	range[0, 1]
voting	range[0, 1]
t	Function: *time()*
t_e	Function: *time()*
punishment rate	1/reputationLevel

Individual TL assessment algorithm

The Individual TL algorithm is implemented according to the formula introduced in Chapter 3. The value defined for the parameters are listed in Table VII.

4.2.4 Implementation of secure communication

After the implementation of the main function blocks, we enabled the communication between each entity in the system by applying client/server mode according to the role of different entities. In order to secure the communication, we applied SSL protocol for the communication, and use OpenSSL (http://www.openssl.org/) for the implementation of SSL-based communication.

Library overview

OpenSSL is an open source library written in C. It provides the implementation of SSL and TLS protocols, as well as some basic cryptographic algorithms and various utility functions. Popular symmetric ciphers such as DES, and public key cryptography such as RSA are available in the crypto-OpenSSL library. Besides, it enables the implementation of digital certificates and authentication functions. Generally, OpenSSL can be either used as

command tool, or library included in the program. Command tool is normally used when generating cryptographic keys, or performing digital signature and authentication. In our implementation, we used OpenSSL command line tool for certificate generation, and used OpenSSL as a library in order to utilize its cryptographic APIs and utility functions.

Table VII Parameter values

Param/Variable	Value/Range
σ	100.0
α	0.5
$\omega 1$	0.5
$\omega 2$	0.3
$\omega 3$	0.2
$N_{c(i,j)}/ N_{m(j,i)}$	100
$N_{m(i,j)}/ N_{m(j,i)}$	50
$N_{i(i,j)}/N_{i(j,i)}$	30
pl (priority)	$TL*(\omega 1 * \theta(N_{c(i,j)} + N_{c(j,i)}) + \omega 2 * \theta(N_{m(i,j)} + N_{m(j,i)}) + \omega 3 * \theta(N_{i(i,j)} + N_{i(j,i)}))$
pu(punishment rate)	1/TL

Prepare private key and certificates:

SSL-based connection is secured by identity verification and data encryption. To start a SSL-based communication, it is necessary to verify each entity's certificates to assure the identity of the other communicator. In our implementation, we used OpenSSL command line tool to generate a self-signed certificate, since it is only for test usage. Because certificates are related to public key cryptography, we firstly generated a private key in order to create and sign a certificate. OpenSSL provides the function and command to generate a RSA-based private key.

```
$ openssl genrsa -out privkey.pem

$ openssl req -new -x509 -key privkey.pem -out cacert.pem -days 1095
```

By the first command, a RSA-based private key is generated and written in *privkey.pem* file. The second command is used to create a self-signed certificate using the generated private key. X.509 is a standard format for public key certificates, and all the key/certificate files are created in *.pem* format.

Setup SSL connection:

SSL connection is setup by firstly initiating the OpenSSL library and packages. In the following functions, *SSL_library_init()* is invoked to initiate the library, and *OpenSSL_add_all_algorithms()* is invoked to load all the algorithms in the library in order to use its APIs in the latter program. *ERR_load_crypto_strings()* and *ERR_load_SSL_strings()* are the functions for loading error handling packages of libraries for crypto and ssl.

```
SSL_library_init();

ERR_load_crypto_strings();

ERR_load_SSL_strings();

OpenSSL_add_all_algorithms();
```

Establish SSL connection:

By finishing loading all the required libraries and packages, we started to set up SSL connections by invoking the following functions. Function *SSL_CTX_new(const SSL_METHOD *method)* creates a SSL_CTX object to establish a SSL enabled connection, and defines if the SSL object is in the client mode or the server mode, by selecting *SSL_METHOD*. OpenSSL provides various options for *SSL_METHOD*, among which we choose *SSLv23_client_method* for client and *SSLv23_server_method* for server. *SSLv23* indicates our connection is compliant to both SSL protocol version2 and version3.

SSL_CTX_load_verify_locations specifies the location of CAfile, which is a library for storing trusted CA certificates, in order to verify if a server's certificate is valid. The certificate matching is based on the subject name, key identifier and the serial number. For I/O operation setup and implementation, we used BIO, which is an I/O abstraction object defined in OpenSSL to simplify I/O operations in an application by encapsulating many I/O details in BIO functions. A BIO can handle SSL connections, unencrypted connections and file I/O. Function *BIO_new_ssl_connect(SSL_CTX)* and *BIO_get_ssl(BIO, SSL*)* are called to initiate a SSL BIO for secure connection and data transmission. *SSL_set_mode(SSL*, SSL_MODE_AUTO_RETRY)* is called to set the SSL connection in *SSL_MODE_AUTO_RETRY* mode. The flag *SSL_MODE_AUTO_RETRY* implies that the SSL connection never bothers the application with retransmission requests if the transport is blocking. It will cause the read/write operations to only return after the handshake and successful completion. Function *BIO_set_conn_hostname(BIO, host_port)* sets the host and its port number during the SSL connection. On the server side, it creates a BIO chain to add up new accepted connection requests, and free them by invoking *BIO_pop(BIO)* and *BIO_free_all(BIO)* after establishing the connections.

```
ctx = SSL_CTX_new(const SSL_METHOD *method);

SSL_CTX_load_verify_locations(SSL_CTX *ctx, const char *CAfile, const char
*CApath)

sbio = BIO_new_ssl_connect(SSL_CTX);

BIO_get_ssl(BIO, SSL*)

SSL_set_mode(SSL*, SSL_MODE_AUTO_RETRY);

BIO_set_conn_hostname(BIO, host_port);
```

By finishing setting up the SSL mode, verification list, and assigning connection host and port, we started a SSL handshake in order to finally establish a SSL connection. In terms of the client side, it firstly starts a connection request by calling *BIO_do_connect(BIO)* on the supplied BIO. Then it invokes *BIO_do_handshake(BIO)* to complete the SSL handshake and establish the SSL connection. On the server side, it firstly sets a BIO chain, at which the new

accepted BIO can be attached to. Function *BIO_do_accept(BIO)* will be invoked twice, where the first time is for creating an accept socket and binding an address, while the second time is used to wait for incoming request or data transmission. After successful SSL connection, *BIO_read(BIO, char*, int)* and *BIO_write(BIO, char*, int)* handles the data transmission between client and server. *BIO_free(BIO)* is invoked to free all the initiated BIO objects.

```
/* Client side */
BIO_do_connect(BIO)
BIO_do_handshake(BIO)
/* Server side*/
BIO_set_accept_bios(BIO, BIO)
BIO_do_accept(BIO)
```

In order to transmit and receive encrypted data/key file correctly and more efficiently, we design the packet format to clarify the basic information of the transmitted data. Fig.14 presents the packet format, which consists of the header and data section. The header section contains file name, file length, and owner/user ID. File name and file length are set for correctly and efficiently reading data, while owner/user ID is set to recognize the file information.

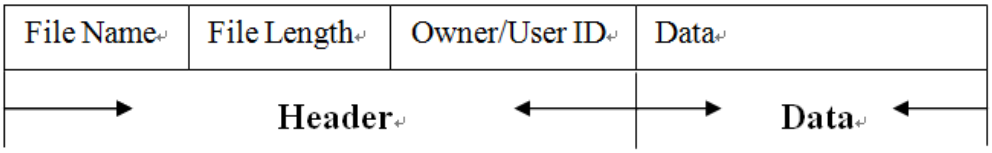

Figure 14 Packet format

Chapter 5 Performance Evaluation

In this chapter, we conducted a number of tests in order to evaluate the performance of the three schemes. We first introduce the test environment including our workstation, encryption key size, and constant values set for system parameters. Next, we provide performance analysis in terms of the computation complexity, data confidentiality, flexibility and key management. Then we present the test cases and test results of the three schemes. Finally, we compare the performance of the three schemes and discuss their usage scenarios.

5.1 Test environment introduction

We implemented and tested the proposed schemes on a workstation with Intel Xeon CPU E31235 and 2-GB RAM, running Ubuntu 12.04 on Oracle VirtualBox. The communication between each entity is based on client/server mode on localhost, and it is secured over SSL protocol by applying OpenSSL library. We applied Advanced Encryption Standard (AES) as the symmetric encryption method. For pairing and elliptic curve based cryptography, we applied Type A pairing because it has the fastest pairing time [79]. The order of the group $\mathbb{G}$ is defined to be 160 bits, and the size of the prime field is defined to be 512 bits. As the size of the prime field is defined, the size of the public key is determined as two times of the field size plus 1 bit.

5.2 Performance analysis

In this section, we provide the performance analysis for the three data access control schemes in cloud computing, in terms of four indicators which are: Computational Complexity, Data Confidentiality, Flexibility, and Scalability.

5.2.1 Computational complexity

Computational complexity is a key indicator of scheme efficiency, and it is an essential factor for applying an access control scheme in practical systems. We evaluate the scheme performance by analyzing their computational complexity of each main operation. Table VIII presents the computational complexity of the three schemes and HASBE scheme proposed in [14].

Table VIII Computational complexity

Operation	Scheme 1	Scheme 2	Scheme 3	HASBE[14]
System Setup	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
User Initiation	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(2M + s)$
Individual TL PK Generation	N/A	$\mathcal{O}(2I)$	$\mathcal{O}(2I)$	N/A
Individual TL SK Generation	N/A	$\mathcal{O}(1)$	$\mathcal{O}(1)$	N/A
Re-encryption Key Generation	$\mathcal{O}(n)$	N/A	$\mathcal{O}(n)$	N/A
Encryption	$\mathcal{O}(1)$	$\mathcal{O}(3w)$	$\mathcal{O}(3w + 2j)$	$\mathcal{O}(2Y + X)$
Decryption	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(j + 1)$	$\mathcal{O}(2Y + 2X)$
Re-encryption	$\mathcal{O}(n)$	N/A	$\mathcal{O}(n)$	N/A
Reputation Evaluation	$\mathcal{O}(m)$	N/A	$\mathcal{O}(m)$	N/A
Individual TL Assessment	N/A	$\mathcal{O}(1)$	$\mathcal{O}(1)$	N/A

Notes:
Scheme1: Reputation and PRE based data access control;
Scheme2: Individual TL and CP-ABE based data access control;
Scheme3: Heterogeneous data access control;
n: the number of authorized users for data access;
m: the number of votes;
I: the maximum number of Individual TLs;
w: the number of TLs specified in the access policy, $w \leq \bar{I}_{tl}$, $\bar{I}_{tl}$ is the maximum numbers of Trust levels;
Y: the number of leaf nodes in an access policy tree;
S: the attribute set;
M: the number of attributes in S;
X: translating nodes of access policy tree;
j: the number of divided key segments

As shown in the table, our schemes are more efficient than HASBE in terms of System Setup and User Initiation, because we constrain the number of attributes and supplement the security level by providing reputation, Individual TL, and the punishment mechanism. Moreover, there are no pairing operations for Encryption in our schemes, thus it makes the computation faster. For Encryption and Decryption, Scheme 1 is the most computationally efficient since the algorithm are not affected by any variables. The computational complexity of Scheme 2, Scheme 3, and HASBE depend on the number of attributes specified in the access policy tree. Scheme 2 and HASBE have the same computational cost if they have the

same number of attributes. Besides the number of attributes, the computational cost of Scheme 3 also depends on the number of symmetric key segments divided by the data owner. The computation of reputation evaluation and TL assessment is quite efficient, since there are neither exponentiations nor pairing operations.

When considering the computational efficiency for each entity in the system, data owners tend to be most lightweight. They require better user experience and are limited by device capability, while service providers (e.g., CSP, RC, etc.) are believed to own adequate system capability. Table VIII shows that Scheme 1 is most computationally efficient for data owners, while the others provide more fine-grained data access and do not require full trust in CSPs.

5.2.2 Data confidentiality

The data confidentiality of the three schemes is evaluated through three factors: cryptographic security, collusion, and punishment mechanism. Cryptographic security is the basic security requirement, since all the schemes are based on cryptographic algorithms. Collusion and punishment mechanism are related to the scheme design in order to encourage better performance and improve system security level.

Cryptographic security

The cryptographic security depends on the arithmetic security of the symmetric encryption algorithm, PRE and CP-ABE. For the symmetric encryption algorithm, we applied AES whose key size is beyond 128bit. It is widely used in multiple cryptographic systems, and is believed to be secure for data encryption. The standard security and master key security of PRE are proved in [77] under the assumption of extended Decisional Bilinear Diffie-Hellman (eDBDH). Additionally, PRE enables non-transitive property to prevent the re-delegation of the decryption rights from, for example, rk_RC->A and rk_A->B to produce rk_RC->B. The arithmetic security of CP-ABE is proved in [12] under the assumption of Decisional BDH (DBDH).

Collusion

The problem of collusion is mainly concerned about the collusion of CSPs and all users in a system, because the RCs and other user authorities are fully trusted under the system model assumptions. CSPs are assumed to be semi-trusted not to disclose users' data, and the plaintext is hidden from the CSPs through symmetric data encryption. Although CSPs do not

disclose stored data nor try to crack to obtain the plaintext, it is possible to collude with users to allow unauthorized access or extension of access right.

For first scheme that depends on reputation and PRE, the problem of collusion between CSP and users is controlled by hiding the plaintext and any secret keys from the CSPs. The only encryption key a CSP has is a re-encryption key for an eligible user. Even though CSP colludes with the user who has been delegated the decryption rights, they can only recover the weak secret g^{a_1}, instead of the secret key of RC. However, since the AES key file is encrypted under the RC's public key and the re-encryption key will not change if the RC and the user's public key pairs remain unchanged, the collusion of CSP and user can break the property of fine-grained data access because the user with decryption right can access all the data from the same data owner. And the CSP can allow the extension of access right even if the data owner informs it to block a specific user.

The second and third scheme that are based on CP-ABE provides higher security by hiding either encryption/decryption operations or secret keys from the CSPs. The access authorization and secret key generation are both controlled by the data owner itself. Moreover, a data owner can re-encrypt the data by modifying the policy tree integrated in the data encryption, instead of updating all public key pairs to manage the collusions, thus reducing the complexity of key management. The third scheme further improves the data confidentiality by dividing the symmetric key into multiple segments, so that a data requester has to recombine all the key segments for decryption. However, the two schemes do not completely eradicate the problem of extending users' access right, since the user revocation is based on the blocking list controlled by CSPs who are semi-trusted.

Punishment mechanism

Punishment Mechanism is applied in the Scheme 1 and Scheme 3 to supplement the data confidentiality. It can reduce the possibility of security problems such as collusion, by monitoring performance and reputation, and carrying out several punishment actions. Driven by the business and profits, CSPs are encouraged to have better performance and dedicate to provide more secure data storage.

5.2.3 Flexibility

The scheme flexibility is evaluated in terms of User Flexibility, which relates to actions or online flexibility of users when dealing with access requests and delegations.

User Flexibility relates to complex operations and online requirements when dealing with data access requests, especially for data owners, because service users expect excellent user experience and less operational complexity. The data owners in the first scheme that depends on PRE and reputation are most flexible, since they do not need staying online to handle the re-encryption and access right authorization. On the contrary, the second scheme requires data owners to handle Individual TL assessment and TL secret key issuing when there is an access request. This mechanism increases the work load of the data owners, but provides higher security level and looses the requirement of trust towards CSPs. Although the third scheme retains the most encryption keys, it provides flexibility for users to balance between operational complexity and security level. The data owners can choose any of the provided mechanisms, and decide whether they are willing to stay online and fully control the access right delegation. This heterogeneous property enables free options to choose an access control mechanism mostly according to their own demands.

5.2.4 Scalability

The essential factors that affect system scalability are key management and user revocation. Table IX lists the number of keys issued or managed by each entity in different schemes.

As shown in the table, scheme 1 that is based on PRE and reputation has the least number of keys that need to be managed. All the entities in the system are responsible to generate and manage their own public key pairs, and the re-encryption algorithm only requires RC's secret key and user's public key. The data owner does not have to handle re-encryption no matter how many access rights it needs to delegate. The public keys in Scheme 2 include the system public key, user's public key and TL public key. The secret keys in Scheme 2 include user's secret key and TL secret key that is issued to an authorized data requester. The reason why scheme 3 has the maximum number of keys is that it enables both PRE and CP-ABE based encryption algorithms, in order to provide flexible access control mechanisms and higher security level. HASBE [14] grants all the key issuing and management to domain authorities that makes the domain authorities the bottleneck of system performance in terms of scalability.

One improvement of the three schemes for reducing key management load is applying reputation and Individual TL evaluation, as well as blacklist for user revocation. This improvement supplements the security level of data confidentiality while decreases the computational cost compared to cryptographic methods. And it monitors all entities' performance and encourages better performances.

Table IX Number of keys owned by different entities

	Scheme 1	**Scheme 2**	**Scheme 3**	**HASBE [14]**
Data Owner	1PK;1SK; 1 Symmetric key	3PK;2SK; 1 Master Key; 1 Symmetric key	4PK; 3SK; 1 Master Key; 1 Symmetric key	1PK; 1SK; 1 Master Key; 1 Symmetric key
User	1PK;1SK; 1 Symmetric key	2PK;2SK; 1 Master Key; 1 Symmetric key	3PK;3SK; 1 Master Key; 1 Symmetric key	1SK; 1 Symmetric key
RC	1PK;1SK; N*Re-encryption key	N/A	1PK;1SK; N*Re-encryption key	N/A
CSP	N/A	N/A	N/A	N/A
Trust Authority	N/A	N/A	N/A	1PK; n*Master Key; m*SK

Notes:
Scheme1: Reputation and PRE based data access control;
Scheme2: Individual TL and CP-ABE based data access control;
Scheme3: Heterogeneous data access control;
N: number of authorized user
n: number of sub-domain authorities
m: number of users in the domain

5.3 Performance Test

In this section, we present the results and analysis of the performance test for the three schemes.

5.3.1 Performance test of Scheme 1

The performance test for Scheme 1 is based on the function blocks of PRE and Reputation Model implemented in Chapter 4. We evaluate the scheme by analyzing the operations conducted by different entities: *Data Owner, User, CSP, RC,* and test their performance for each operation. Table X lists the main operations and computation efforts managed by different entities in the system.

The *Data Owner* is only responsible for its public key pair generation and encryption including AES and PRE encryption. The computational cost for AES depends on the size of the underlying data, and it is inevitable in any cryptographic method. The PRE encryption requires 2 exponentiations, and the key generation requires 1 pairing and 1 exponentiation. The computation for a data owner is quite lightweight since it does not need to handle re-encryption or key management no matter how many authorizations of data access right need to be issued. The *User* who requests to access data in the *CSP* only needs to decrypt data and key file using its own secret key, no matter which owner the encryption key comes from.

The CSP in this scheme is responsible for data re-encryption and user revocation. For data re-encryption, it takes 1 pairing, and the computational effort is linear to the number of users who are authorized to access data. User revocation does not require any computational effort. Instead, it does the database lookup and blocks the user's access right according to the data owner's demand.

The *RC* is a trusted third-party responsible for checking data's access policy and a user's reputation level, in order to determine if the user has the right to access the required data. If it is the case, the *RC* then issues the re-encryption key to the *CSP* using the authorized user's public key. The computational effort consists of two parts: re-encryption key generation and reputation evaluation. Re-encryption key generation requires 1 exponentiation for every request, so that the computation is linear to the number of users who request data access at any time point. Reputation evaluation depends on the number of users who request for data access, and the number of the other entities that provide votes.

Table X Operation and computations in Scheme 1

Role	Operations	Computations
Data Owner	Public Key pair Generation	1Pairing + 1Exp
	Encryption	2Exp
User	Public Key pair Generation	1Pairing + 1Exp
	Decryption	1Exp
CSP	Re-encryption	n Pairing
RC	Public Key pair Generation	1Pairing + 1Exp
	Re-encryption Key Generation	nExp
	Reputation Generation	(mMult + mExp)K

Notes:
Exp: Exponentiation;
Pairing: Bilinear Pairing;
Mult: Multiplication;
n: Number of authorized users for data access;
m: Number of votes;
K: the total number of users who request data access;

Fig.15 presents the execution time of each operation conducted by different entity, in the cases of 128-bit, 192-bit and 256-bit sized AES keys. The reputation evaluation contains 10000 user votes. As shown in Fig.15, the algorithm of PRE encryption, decryption and Delegation (re-encrypt key generation) consumes less computational power, because they do not contain pairing operations. We can observe that the Re-encryption handled by CSP is the most time consuming process because it contains pairing operations, which are the most expensive computations in the PRE algorithm. Additionally, the execution time of PRE algorithm over AES symmetric key does not differ much with the tested three-sized AES keys. This fact could greatly benefits data owners to choose a long symmetric key to ensure high level of data security. The execution time of Delegation, which herein refers to the operation of issuing re-encryption key *rkRC->u* to the authorized user, also remains the same with different sized AES keys, since it only operates on the private key of RC and the public key of the user. The reputation evaluation time is not influenced by AES key size.

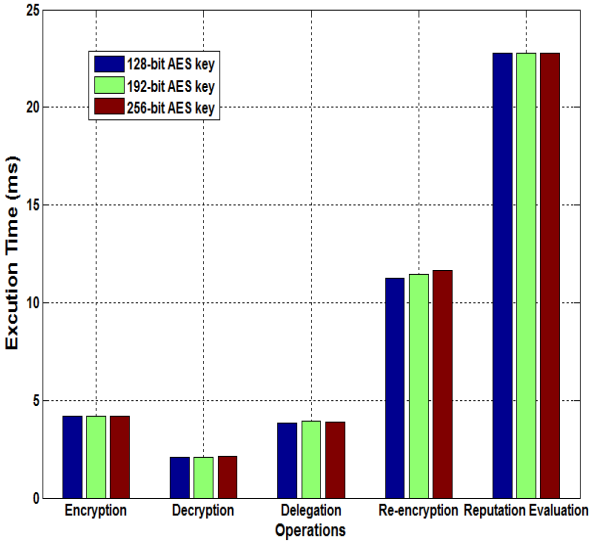

Figure 15 Execution time of scheme operations

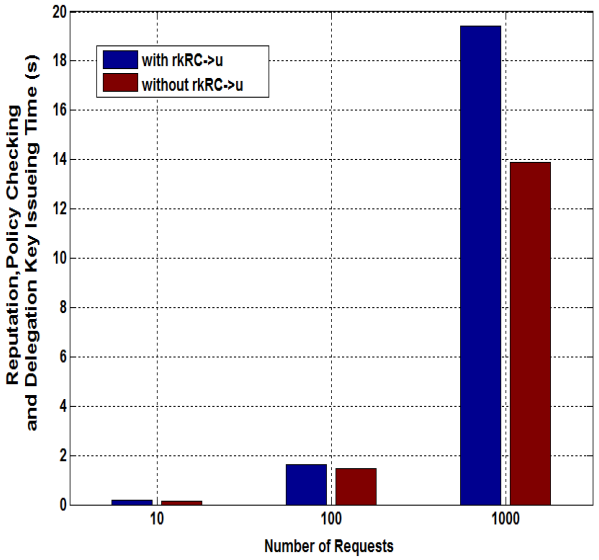

Figure 16 Time comparison of rk_RC->u and NON rk_RC->u key generation for old users

One improvement we have in our scheme for PRE is to skip the re-encryption key generation process at RC, if the key of the user has been already generated before and the user's information remains unchanged. Fig.16 shows the request processing time at RC, including data policy checking reputation evaluation and re-encryption key generation, in two cases. In the first case, the RC generates a re-encryption key for every request if the requester satisfies the policy and reputation level. In the second case, RC directly fetches the already generated re-encryption key from the granted user's record. As shown in Fig.16, the higher the number of requests from the users who have the records at RC, the more efficiency improvement our scheme can achieve. In practice, the user could access cloud services multiple times. Thus, utilizing the existing re-encryption keys greatly helps the RC to improve its efficiency and capacity.

We compare our scheme with the personal health records sharing scheme over cloud proposed in [80] based on ABE (combination of Multi-Authority ABE and Key-Policy ABE). Table XI presents the time comparison between our scheme and [80]. The encryption and decryption time (containing 50 attributes) in [80] is 264ms and 25ms respectively, which is much longer than that in our scheme. Although the experiment workstations are different, the apparent time differences can still indicate that our scheme is more efficient. In addition, sophisticated key management and user revocation are replaced by various access policies regarding user reputation, which has much less time consuming computations.

Table XI Operation time comparison with scheme in [80]

Time	Our Scheme (tested on a workstation with 3.2 GHz processor, 256-bit AES key and 512-bit prime field)	Records Sharing scheme in [80] (tested on a workstation with 3.4 GHz processor, 256-bit AES key and 512-bit prime field)
Key Encryption	4.2ms	264ms
Key Decryption	2.1ms	25ms
User Revocation	Included in Reputation and Data policy checking	32ms
Re-Encryption	11.7ms	N.A
Reputation Evaluation	22.7ms (for 10000 votes)	N.A
total sum of operations	40.7ms	321ms

5.3.2 Performance test of Scheme 2

We evaluate the scheme by analyzing the operations conducted by different entities: Data Owner, User, CSP, and test their performance for each operation. Table XII lists the main operations and computation efforts managed by different entities in the system.

The system setup, including Master Key (MK) and system Public Key (PK) generation, can be conducted in User Agent, or other trusted authorities in the system. The MK and PK are global parameters that are shared between users to initiate their own public key pairs. And they do not change with either the number of users in the system, or the specified maximum Individual TLs for access policy.

Table XII Operation and computations

Role	Operations	Computations
User Agent	Master Key Generation	1Exp
	System Public Key Generation	1Pairing
Data Owner	CSP Reputation Evaluation	(mMult + mExp)K
	Public Key pair Generation	2Exp
	Individual TL Public Key Generation	I*2Exp
	Issue TL Secret Key	1Exp
	Encryption	w*3Exp
	Trust Assessment	K*4Exp
User	CSP Reputation Evaluation	(mMult + mExp)K
	Public Key pair Generation	2Exp
	Decryption	2Pairing
CSP	Checking user ID validity and blacklist	User Data Search

Notes:
Exp: Exponentiation;
Pair: Bilinear Pairing;
w: the number of TLs specified in the access policy, $w \leq \bar{I}_{tl}$, $\bar{I}_{tl}$ is the maximum number of Trust levels;
I: the maximum number of Individual TLs;
m: Number of votes;
K: the total number of users who request data access;

In this scheme, either Data Owner or User who requests for data service can firstly evaluate the CSP's reputation through the Reputation Evaluation Model, in order to ensure if the CSP is trustworthy enough to provide data storage service. Data Owner is responsible for encryption, verifying requester's eligibility through individual trust assessment, and issuing a TL based secret key. Encryption consists of data encryption through AES, and CP-ABE encryption. The computational cost for AES encryption depends on the size of the underlying data, and it is inevitable in any cryptographic method. For CP-ABE encryption, the data owner firstly generates the TL public key *PK_TL*, which covers each of the TLs specified in the system. The computational cost for generating *PK_TL* is I*2Exp, in which I

is the maximum number of TL. Next, the data owner encrypts the AES key using *PK_TL*, according to the required TL specified in the access policy. The computational cost is w*3Exp, in which w is the number of TLs enabled in the access policy. For verifying a requester's access right, the data owner evaluates the requester's individual TL through the Trust Assessment Model, which contains 4 exponentiations for each evaluation. The TL secret key *SK_TL* is issued by the data owner, and implies the requester's individual TL. The user who requests for data can use the issued *SK_TL* to decrypt the AES key file. The CP-ABE Decryption algorithm contains two pairing operations no matter how complex the access policy is, because the algorithm will conduct the decryption algorithm only if the TL in *SK_TL* meets one of the TL attributes in the access policy.

CSP is computationally lightweight since it does not need to carry any data encryption or key issuing. Besides of the data storage, the CSP is responsible for checking all users' identities and user revocation by searching through the blacklist.

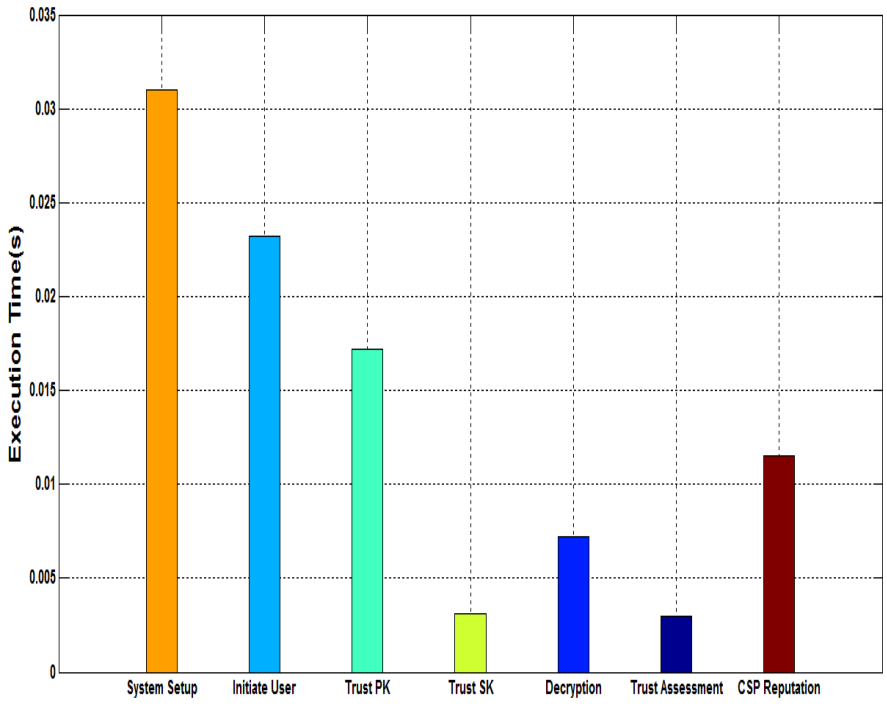

Figure 17 Execution time of operations in Scheme 2

In the performance test, we specify the Individual TLs as attributes and divide them into 5 levels ($\bar{I}_{tl} \leq 5$). Fig.17 presents the execution time of Setup, User Initiation including user's public key pair generation, Individual TL public key and secret key generation, Decryption, Trust Assessment and CSP's reputation evaluation. As introduced in Table XII, the required

TLs in access policy have no impact on the performance of the above operations presented in Fig.17. Since we specify the maximum TL in the system, the TL public key *PK_TL* generation takes about 17ms, and issuing TL secret key *SK_TL* is less than 5ms. In various applications, *PK_TL* generation process can be different depending on the number of specified levels, as shown in Fig.18. The execution time of *SK_TL* generation stays constant and it is about 3ms, which implies that the *SK_TL* issuing process should be very efficient. Moreover, the decryption time is about 7ms. Trust Assessment is conducted by the data owner, and it costs around 3ms to evaluate the TLs for 10000 users. The reputation evaluation towards CSP contains 100000 votes and took about 10ms, which is reasonable for the system performance.

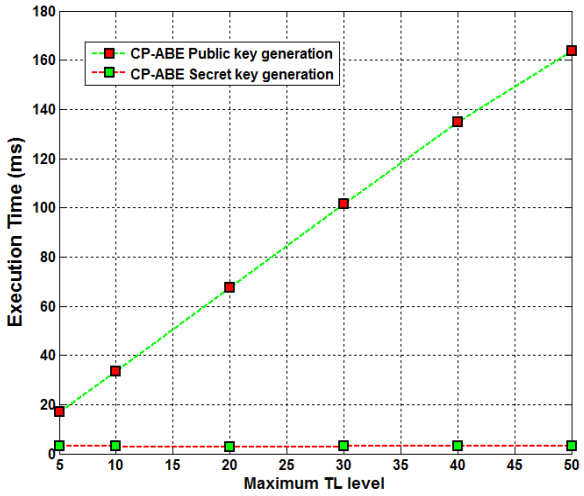

Figure 18 Execution time of PK_TL and SK_TL with different maximum TL levels

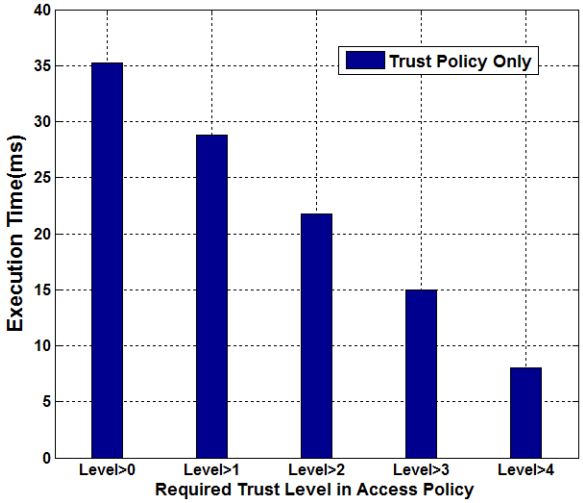

Figure 19 Execution time of encryption with different required TL

Fig.19 shows the performance of Encryption algorithm with different required TLs. It illustrates that the higher the required trust level, the less time the Encryption spends, because there are less authorized access conditions to be enabled.

5.3.3 Performance test of Scheme 3

Scheme 3 provides a flexible way for the data owner to choose either PRE/Reputation based or CP-ABE/TL based data access control mechanism. In the performance evaluation, we firstly provide four mechanisms for users to choose from. Next, we test the performance of each operation in both PRE/Reputation based and CP-ABE/TL based access control. Table XIII presents the four mechanisms that are enabled in the scheme.

Table XIII Mechanisms for heterogeneous data access control

	Access Control Method	Key Division
Mechanism 1	Only PRE and Reputation Evaluation	$K_0 = n_0$; $K_1 = n_1$; …… $K_n = n_n$; K_{n+1}= null ($n_0+n_1+\ldots n_n$=size of K; n is the number of RCs)
Mechanism 2	Only CP-ABE and Individual TL Assessment	K_0= null; K_1 = K
Mechanism 3	Both PRE-Reputation and CP-ABE-TL	$K_0 = n_0$; $K_1 = n_1$; …… $K_n = n_n$; K_{n+1}=m ($n_0+n_1+\ldots n_n$+m=size of K; n is the number of RCs)
Mechanism 4	Either PRE-Reputation or CP-ABE-TL	$K_0 = n_0$; $K_1 = n_1$; …… $K_n = n_n$; K_{n+1} = K ($n_0+n_1+\ldots n_n$=size of K; n is the number of RCs)

Notes:
K: the size of AES key;
$\mathbf{K_n}$: nth key segment for PRE and Reputation based access control
$\mathbf{K_{n+1}}$**:** key segment for CP-ABE and TL based access control

In our performance test, we employed one RC for reputation management, and proved that the performance is negligibly affected by the number of RCs. As shown in Table XIII, Mechanism 1 and Mechanism 2 enable only one of the two access control methods which are reputation based or Individual TL based method. Therefore, data owner does not need to divide the AES key. In Mechanism 3, the AES key is divided into two segments K0 and K1, in order to enable both access control methods for dual data protection. Mechanism 4 provides an option to use either of the two access control methods, so that a data requester can get either K0 or K1 to decrypt the data. Fig.20 shows the packet format of encrypted key file. The sequence number indicates the position of the key segment, in order to correctly recombine K0 and K1.

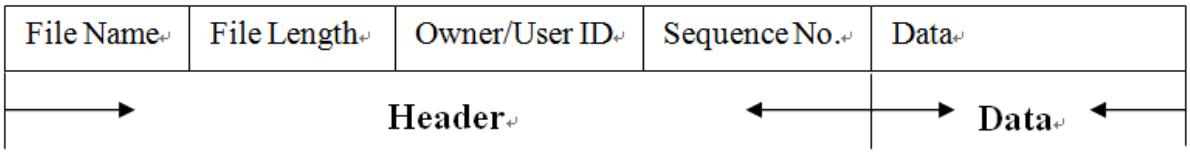

Figure 20 Packet format

Fig.21 shows the execution time of CP-ABE setup, CP-ABE key pair generation, PRE key pair setup and generation, TL public/secret key generation, and Re-encryption key generation. The performance of the operations is not affected by either the user's preference of key division or the required individual Trust Level. In the performance test, we assumed 5 trust levels, and the TL public key generation process can vary with different number of maximum trust levels, as shown in Fig.18.

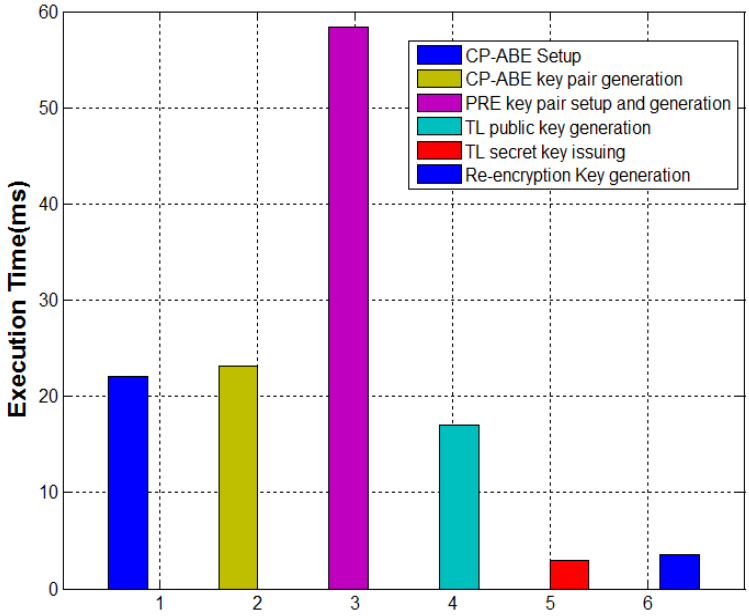

Figure 21 Execution time of operations in Scheme 3

Fig.22 shows the execution time of reputation evaluation and policy checking which is conducted in RC, and trust assessment that is processed by the data owner. The performance of reputation evaluation depends on the number of votes received by RC, and the execution time of Individual TLs depends on the number of access requests received by the data owner. The computational cost for trust assessment is quite low, which reduces the computation load of data owner.

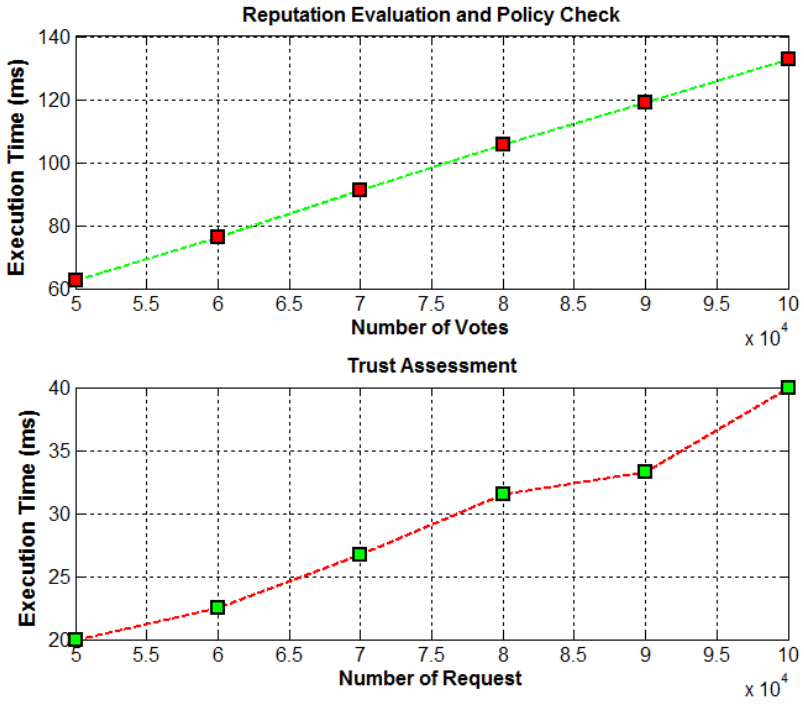

Figure 22 Execution time of trust assessment, reputation evaluation and policy check,

For the performance evaluation, we test three different sized AES keys: 128-bit, 192-bit and 256-bit. Fig.23 illustrates that the AES key size has little effect on the performance of CP-ABE encryption, as well as the decryption which takes around 6.50ms in the test, shown in Fig.24. For different required Individual TLs, the encryption time varies because different numbers of authorized levels are enabled in the access policy. The higher the required trust level is, the less time the encryption process spends.

Fig.25 presents the performance of the PRE process. We can observe that the PRE operations are also not clearly affected by the size of (partial) AES key. This fact could benefit the data owners to choose a long symmetric key if they need high level of data security and the selection of access control mechanisms can depend on user's requirements in terms of security and efficiency, instead of the size of key segment. We can also observe that the computational cost of CP-ABE is higher than that of PRE algorithm, because CP-ABE carries more exponentiation and pairing operations.

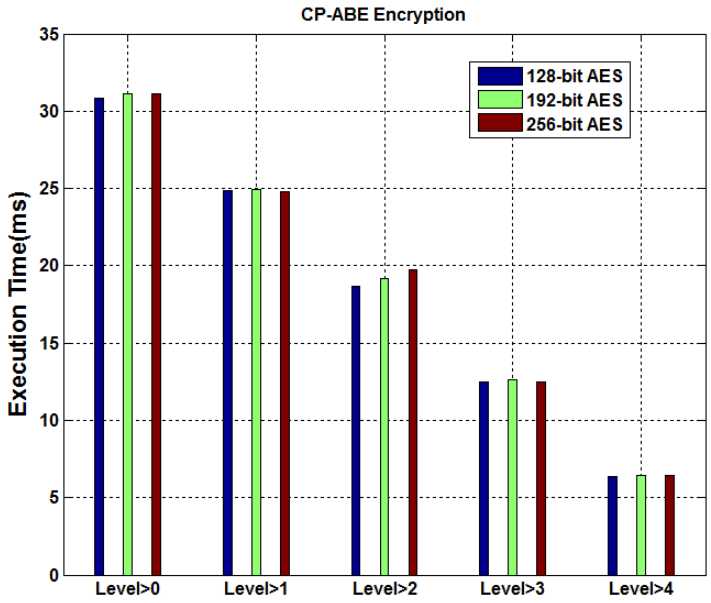

Figure 23 CP-ABE encryption time of AES keys

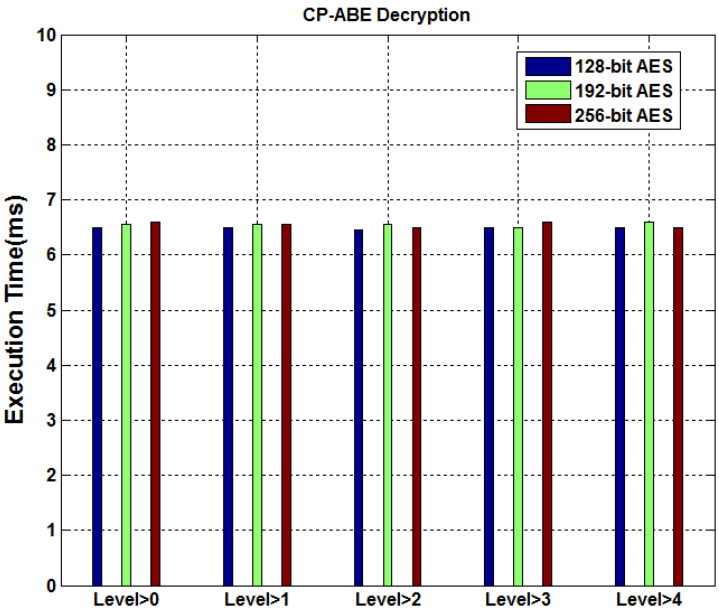

Figure 24CP-ABE decryption time of AES keys

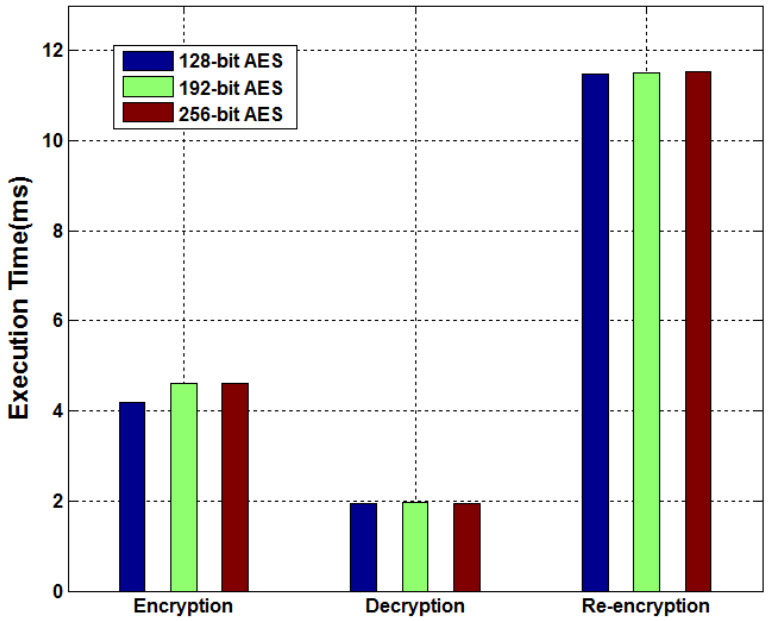

Figure 25 Execution time of PRE operations

5.4 Performance comparison and user scenario discussion

Upon the performance test and analysis in previous sections, we provide detailed comparisons between the three schemes in Table XIV, and discuss their suitable use scenarios. We can observe from Table XIV that all three schemes apply AES as symmetric cryptographic method for data encryption, and blacklist for user revocation to reduce the computational cost. Scheme 2 and Scheme 3 which apply CP-ABE as cryptographic algorithm have higher level of data confidentiality, because they require the data owners to issue the private key by themselves, instead of depending on the CSPs or the RCs.

For Scheme 1, most of the computational cost is distributed in RC and CSP for re-encryption key generation and ciphertext re-encryption. This design frees data owners from heavy computations, and there is no online requirement for data owners to deal with access requests. PRE and reputation-based evaluation also increase the system scalability by simplifying key management, in which a user holds one key pair and only RC is responsible for managing the re-encryption keys. In terms of the system model, the RCs should be fully trusted in order to insure that the access rights can be delegated only to eligible users. However, it is difficult and risky to put full trust on one entity in a public cloud, which decreases the level of data confidentiality of Scheme 1, compared to the other two schemes. Scheme 1 is suitable for the private cloud environment, or data storage system inside an enterprise, because the infrastructure and authority is trusted in this case.

Table XIV Comparison table

	Techniques	Computation Distribution	Data Confidentiality	System Model Assumption	User Revocation	Scalability
Scheme 1	AES, PRE, Reputation Evaluation	Mostly in RC and CSP	Average	RC fully-trusted, CSP semi-trusted	Blacklist	High
Scheme 2	AES, CP-ABE, Individual Trust	Mostly in Data Owner	High	CSP semi-trusted	Blacklist	High
Scheme 3	AES, PRE, CP-ABE, Reputation Evaluation, Individual Trust	Mechanism Dependable	High (specific security level can be chosen by data owner)	RC fully-trusted, CSP semi-trusted	Blacklist	Medium

Scheme 2 has higher data confidentiality level, because it requires the data owners to assess the requesters' trust levels and issue the ABE private keys by themselves. The CSPs are only responsible for user ID verification, and providing data services. There is no requirement for fully trusted entity for key or access right authorization, comparing to Scheme 1. Applying user blacklist can increase the system efficiency and scalability by simplifying the user revocation process, and reducing the operations for key updating and ciphertext re-encryption. However, CP-ABE brings heavier computational load than PRE algorithm. And most computations, such as encryption and ABE key generation, are conducted by data owners, which can affect user experience. Additionally, data owners are required to be online in order to process data requests and issue private keys to eligible users. Scheme 2 is suitable for the public cloud environment, since it enables higher confidentiality and security level. It can also be applied to applications for social networks. For example, in a professional social network, a user can decide which company is eligible to access his/her resume by evaluating the company's individual trust level.

Comparing to Scheme 1 and Scheme 2, Scheme 3 provides more options in terms of cryptographic algorithms used for data access control. It enables more flexibility for users to

choose a balance point between computational complexity and data confidentiality. Different from Scheme 1, there could be multiple RCs to evaluate a user's reputation, and manage re-encryption keys. This design improves the security level of PRE and reputation based data access control scheme, because a user needs to be authorized by all RCs to get a complete AES key for decryption. However, the flexibility of user's option can lead to complexity of system operation and key management. It is difficult for a CSP to manage multiple AES key segments for a user, especially when there are a huge number of users in the cloud environment. Because Scheme 3 applies both PRE and CP-ABE algorithms, it requires key pairs for both cryptographic algorithms, which increases the total number of keys in the system and makes the key management more sophisticated. Scheme 3 can be deployed in a public cloud environment because of its high security level and flexibility. For example, it can be applied in a public health-care system, in which a patient can delegate the access right of the personal information to his/her doctors by choosing PRE and reputation based control mechanism. And the patient can choose CP-ABE and Individual TL based mechanism to issue the access right to other research organizations or third parties.

Chapter 6 Conclusions and Future Work

In this chapter, we present the conclusions of this thesis, and propose several improvements for the future works.

6.1 Conclusions

To protect the data and privacy from disclosure, a number of schemes have been proposed for data access control in cloud computing. Before applying an access control scheme in a practical system, it is indispensable to evaluate its performance in various aspects, such as efficiency and flexibility. In this thesis, we addressed the implementation and evaluation of three schemes for data access control in cloud computing. We implemented the reputation and Individual TL assessment algorithms, and integrated them with the cryptographic algorithms such as PRE and CP-ABE to develop data access control schemes. Furthermore, we implemented the SSL-based communication to enable the secure data transmissions.

Based on the implementations, we conducted a number of performance tests for the three schemes. We then presented the results of the performance evaluation, the analysis and comparison on several properties, such as data confidentiality, computational complexity, flexibility and key management.

According to the performance evaluation, the computational efficiency is an important factor for designing a data access control scheme, especially when deploying the scheme in practical systems. Key management is a vital issue for cryptography based systems, especially for the system scalability. It can influence whether the enterprises are willing to adopt the access control scheme, because key management can sometimes very complicated and resource consuming. Reputation and individual trust can be an effective method to control data access in cloud computing, and also for choosing CSPs with better performances. It can be applied in combination with cryptographic algorithms, and help to reduce the computational cost.

6.2 Future work

After the current implementation and performance evaluation, we realize that there can be more effort relating to system implementation, scheme evaluation, and further deployment into practical applications. The future work may include:

Enrich the system implementation: The current implementation contains one data owner, one user, and one CSP to form a basic system that applies the data access control schemes. Therefore, we can extend the network scale that contains more users and CSPs, as well as more RCs to support multi-dimensional access authorization in a system. This improvement can make our implementation close to practical systems, and enables more comprehensive performance evaluation.

Integrate and encapsulate the function blocks: We could further integrate and encapsulate the current function blocks to develop an effective tool box for future implementation and development.

Provide real network data for reputation and Individual TL evaluation: The current implementation and evaluation is based on the already collected data, such as users' voting and network monitoring results. In the future work, we can collect users' data, and monitor user behavior in a real network in order to evaluate their real reputation and trust level.

More comprehensive performance evaluation: Based on the above improvements for system implementation and data collection, we can provide more comprehensive evaluation towards the proposed access control schemes. For example, we can develop a test platform which can be more generally applied for implementation and evaluation of data access control schemes. This platform can contain a toolbox which consists of function blocks of some common cryptographic algorithms, such as PRE, CP-ABE. Moreover, it can be applied to monitor network flow to test communication cost, to simulate simultaneous requests to test system throughput and scheme efficiency, and other comprehensive performance evaluations.

Deployment in practical systems: We can deploy the data access control schemes in practical systems or applications. Thus we could monitor the performance, and adjust the schemes to practical cases.

Business model and legal aspects: In order to deploy a data access control scheme in practical systems or applications, it is sometimes inevitable to develop a corresponding business model. The access control services enabled by the scheme need to be rational and feasible in the aspect of business. For legal issues, we firstly need to be clear about the rules or laws regarding to data or privacy protection in a specific use scenario before deploying the data access control scheme, in order not to violate any data or privacy regulations.

References

[1] The NIST Definition of Cloud Computing, http://www.nist.gov/itl/cloud/ .

[2] L.Popa, M.Yu, Y.K.Steven, S.Ratnasamy, I.Stoica, "CloudPolice: taking access control out of the network", in Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Article No.7, New York, 2010, ISBN: 978-1-4503-0409-2.

[3] OAuth, http://oauth.net/.

[4] J.M.Alcaraz Calero, N.Edwards, J.Kirschnick, L.Wilcock, M.Wray, "Towards a Multi-tenancy Authorization System for Cloud Services", IEEE Security & Privacy, 2010, Volume 8, Issue 6, IEEE P.48, ISSN 1540-7993.

[5] Zheng Yan, "Chapter 4: Trust Management in Mobile Environments, Trust Management in Mobile Environments – Usable and Autonomic Models", IGI Global, Hershey, Pennsylvania, USA, 2013. DOI: 10.4018/978-1-4666-4765-7, ISBN13: 9781466647657, ISBN10: 1466647655, EISBN13: 9781466647664.

[6] M.Alhamad, T.Dillon, E.Chang, "SLA-Based Trust Model for Cloud Computing", 2010 13th International Conference on Network-Based Information Systems, Sept 2010, IEEE P.321, E-ISBN 978-0-7695-4167-9.

[7] W.J.Li, L.D.Ping, X.Z.Pan, "Use trust management module to achieve effective security mechnisms", 2010 International Conference on Electronics and Information Engineering, Aug 2010, IEEE P.v1-14, E-ISBN 978-1-4244-7681-7.

[8] H.Sato, A.Kanai, S.Tanimoto, "A Cloud Trust Model in a Security Aware Cloud", 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet, July 2010, IEEE P.121, E-ISBN 978-0-7695-4107-5.

[9] A.Sahai, B.Waters, "Fuzzy identity-based encryption", in Proceedings of the 24th annual international conference on Theory and Applications of Cryptographic Techniques, Berlin 2005, Springer P.457, ISBN 3-540-25910-4 978-3-540-25910-7.

[10] V.Goyal, O.Pandey, A.Sahai, B.Waters, "Attribute-based encryption for fine-grained access control of encrypted data", in Proceedings of the 13th ACM conference on Computer and communications security, New York 2006, ACM P.89, ISBN 1-59593-518-5.

[11] J.Bethencourt, A.Sahai, B.Waters, "Ciphertext-policy attribute based encryption", 2007 SP'07.IEEE symposium on Security and Privacy, May 2007, IEEE P.321, ISBN 0-7695-2848-1.

[12] S.Müller, S.Katzenbeisser, C.Eckert, "Distributed Attribute-Based Encryption," in International Conference on Information Security and cryptology, Berlin 2009, Springer P.20, ISBN 978-3-642-00729-3.

[13] S.C.Yu, C.Wang, K.Ren, W.J.Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing", in Proceedings of the 29th conference on Information communications, San Diego March 2010, IEEE P.1, ISBN 978-1-4244-5836-3.

[14] Z.G.Wan, J.E.Liu, R.H.Deng, "HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing", IEEE Transactions on Information Forensics and Security, Oct 2011, Volume 7, Issue 2, IEEE P.743, ISSN 1556-6013

[15] F. K. Hussain, E. Chang, "An overview of the interpretations of trust and reputation," The Third Advanced International Conference on Telecommunications, Morne AICT 2007, IEEE P.30, ISBN 0-7695-2843-0.

[16] A.Dasgupta, A.Prat, "Reputation and asset prices: A theory of information cascades and systematic mispricing," Manuscript, London School of Economies, Jan.2005.

[17] A.Abdul-Rahman, S Hailes, "Supporting trust in virtual communities," in Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Jan.2000.

[18] H.Sato, A.Kanai, S.Tanimoto, "A Cloud Trust Model in a Security Aware Cloud", 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet, Seoul July.2010, IEEE P.121, ISBN 978-1-4244-7526-1.

[19] J.H.Yao, S.P.Chen, W.Chen, D.Levy, J.Zic, "Accountability as a Service for the Cloud", 2010 IEEE International Conference on Services Computing (SCC), Miami July.2010, IEEE P.81, ISBN 978-1-4244-8147-7.

[20] P.S.Pawar, M.Rajarajan, S.K.Nair, A.Zisman, "Trust Model for Optimized Cloud Services", in Proceedings of 8th IFIP WG11.11 International Conference, IFIPTM 2014, Singapore July.2014, Springer P.97.

[21] S.K.Prajapati, S.Changder, A.Sarkar, "Trust Management Model for Cloud Computing Environment", in Proceedings of the International Conference on Computing, Communication and Advanced Network (ICCCAN 2013), India March.2013, P.1-5.

[22] S.M.Habib, S.Ries, M.Muhlhauser, "Towards a Trust Management System for Cloud Computing", 2011 IEEE 10th International Conference on Trust, Security and Privacy

in Computing and Communications (TrustCom), Changsha Nov.2011, IEEE P.933, ISBN 978-1-4577-2135-9.

[23] Z.Yan, X.Y.Li, R.Kantola, "Personal Data Access Based on Trust Assessment in Mobile Social Networking", 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, Beijing Sept.2014, IEEE P.989, DOI 10.1109/TrustCom.2014.131.

[24] S.M.Habib, S.Ries, M.muhlhauser, "Cloud Computing Landscape and Research Challenges regarding Trust and Reputation", 2010 7th International Conference on Computing (UIC/ATC), Xian Oct.2010, IEEE P.410, ISBN 978-0-7695-4272-0.

[25] A.Bradai, W.Ben-Ameur, H.Afifi, "Byzantine Resistant Reputation-based Trust Management", 2013 9th International Conference on Collaborative Computing: Networking, Applications and Worksharing, Austin Oct.2013, IEEE P.269.

[26] A.Koneru, K.Venugopal Rao, "Reputation Management in Distributed Community Clouds", International Journal of Advanced Research in Computer and Communication Engineering, Dec.2012, Vol.1, Issue 10, ISSN 2278-1021.

[27] S.P.Muralidharan, V.V.Kumar, "A Novel Reputation Management System for Volunteer Clouds", 2012 International Conference on Computer Communication and Informatics, Coimbatore Jan.2012, IEEE P.1, ISBN 978-1-4577-1580-8.

[28] C.S.Zhu, H.Nicanfar, V.C.M.Leung, L.T.Yang, "An Authenticated Trust and Reputation Calculation and Management System for Cloud and Sensor Networks Integration", IEEE Transactions on Information Forensics and Security, Volume 10, Issue 1, Oct.2014, IEEE P.118, ISSN 1556-6013.

[29] Z.Yan, X.Y.Li, R.Kantola, "Controlling Cloud Data Access Based on Reputation", Mobile Networks and Applications, Springer, March 2015, Springer, ISSN 1572-8153.

[30] Q.T.Wu, X.L.Zhang, M.C.Zhang, Y.Lou, R.J.Zhang, W.Y.Wei, "Reputation Revision Method for Selecting Cloud Services Based on Prior Knowledge and a Market Mechanism", The Scientific World Journal, Volume 2014, Article ID 617087, ISSN 1537-744X.

[31] M.Wang, G.L.Wang, J.Tian, H.W.Zhang, Y.J.Zhang, "An Accurate and Multi-faceted Reputation Scheme for Cloud Computing", The 11th International Conference on Mobile Systems and Pervasive Computing (MobiSPC'14), Procedia Computer Science 2014, Volume 34, P.466, ISSN 1877-0509.

[32] D.F.Ferraiolo, D.R.Kuhn, “Role-Based Access Control”, 15th National Computer Security Conference, Baltimore Oct.1992, P.554.

[33] E.Bertina, P.A.Bonatti, E.Ferrari, “TRBAC: A temporal role-based access control model”, Journal of ACM Transactions on Information and System Security, Volume 4, Issue 3, Aub.2001, P.191.

[34] J.B.D.Joshi, E.Bertino, U.Latif, A.Ghafoor, “A generalized temporal role-based access control model”, IEEE Transactions on Knowledge and Data Engineering, Volume 17, Issue 1, Jan.2005, P.4, ISSN 1041-4347.

[35] D.G.Yu, “Role and Task-based Access Control Model for Web Service Integration”, Journal of Computational Information Systems, Volume 8, Issue 7, P.2681, 2012.

[36] M.Barati, S.K.Mohammad, S.Lotfi, A.Rahmati, “A New Semantic Role-based Access Control Model for Cloud Computing”, the 9th International Conference on Internet and Web Applications and Services, Paris July.2014, ISBN 978-1-61208-361-2.

[37] Z.Tang, J.Wei, A.Sallam, K.Li, R.X.Li, “A new RBAC based access control model for cloud computing”, in Proceeding of the 12th International Conference on Advances in Grid and Pervasive Computing, Berlin 2012, Springer P.279, ISBN 978-3-642-30766-9.

[38] S.Y.Na, S.H.Cheon, “Role Delegation in Role-Based Access Control”, in Proceeding of the 5th ACM workshop on Role-based access control, NewYork 2000, ACM P.39, ISBN 1-58113-259-X.

[39] G.Y.Lin, Y.Y.Bie, M.Lei, “Trust Based Access Control Policy in Multi-domain of Cloud Computing”, Journal of Computers, Volume 8, Number 5, May.2013, ISSN 1796-203X.

[40] Gitanjali, S.S.Sehra, J.Singh, “Policy Specification in Role based Access Control on Clouds”, International Journal of Computer Applications, Volume 75, Number 1, 2013, ISSN 0975 - 8887.

[41] Natarajan Meghanathan, “Review of Access Control Models For Cloud Computing”, in Proceedings of the 3th International Conference on Computer Science, Engineering & Applications , ICCSEA, India May.2013, P.77, DOI : 10.5121/csit.2013.3508.

[42] A.Sahai, B.Waters, “Fuzzy Identity Based Encryption”, in Proceedings of 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Denmark May.2005, Springer P.457, ISBN 978-3-540-32055-5.

[43] V.Goyal, O.Pandey, A.Sahai, B.Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in Proceedings of the 13th ACM conference on

Computer and communications security, New York 2006, ACM P.89, ISBN 1-59593-518-5.

[44] A. Beimel, 1996, "Secure Schemes for Secret Sharing and Key Distribution," PhD Thesis, Israel Institute of Technology, Technion, Haifa, Israel, 115p.

[45] M.Chase, "Multi-authority Attribute based Encryption", in Proceeding of the 4th Conference on Theory of Cryptography, Berlin 2007, Springer P.515, ISBN 978-3-540-70935-0.

[46] C.J.Wang, J.F.Luo, "A Key-policy Attribute-based Encryption Scheme with Constant Size Ciphertext", the 8th International Conference on Computational Intelligence and Security, Guangzhou Nov.2012, IEEE P.447, ISBN 978-1-4673-4725-9.

[47] H.Y.Yan, X.Li, J.Li, "Secure Personal Health Record System with Attribute-Based Encryption in Cloud Computing", 9th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Guangdong Nov.2014, IEEE P.329, DOI 10.1109/3PGCIC.2014.138.

[48] Z.Q.Lv, J.L.Chi, M.Zhang, D.G.Feng, "Efficiently Attribute-Based Access Control for Mobile Cloud Storage System", IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Beijing Sept.2014, IEEE P.292, DOI 10.1109/TrustCom.2014.40.

[49] Shucheng Yu, Cong Wang, Kui Ren, Wenjing Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in Proceedings of the 29th conference on Information communications, San Diego March.2010, IEEE P.1, ISBN 978-1-4244-5836-3.

[50] J. Bethencourt, A. Sahai, B. Waters, "Ciphertext-policy attribute based encryption," in Proceedings of SP'07.IEEE Symposium on Security and Privacy, Washingtong May.2007, IEEE P.321, ISBN 0-7695-2848-1.

[51] A.Lewko, B.Waters, "Decentralizing Attribute-Based Encryption", the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn May.2011, Springer P.568, ISBN 978-3-642-20465-4.

[52] M.Horvath, "Attribute-Based Encryption Optimized for Cloud Computing", the 41st International Conference on Current Trends in Theory and Practice of Computer Science, Czech, Springer P.566, ISBN 978-3-662-46078-8.

[53] D.Y.Xu, F.Y.Luo, L.Gao, Z.Tang, “Fine-grained document sharing using attribute-based encryption in cloud servers”, the 3rd International Conference on Innovative Computing Technology, London Aug.2013, IEEE P.65, ISBN 978-1-4799-0047-3.

[54] Zhiguo Wan, Jun’e Liu, R.H.Deng, “HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing,” IEEE Transactions on Information Forensics and Security, Volume 7, Issue 2, IEEE P.743, Oct.2011, ISSN 1556-6013.

[55] L.Z.Xiong, Z.Q.Xu, “Re-encryption security model over outsourced cloud data”, 2013 International Conference on Information and Network Security, Beijing Nov.2013, IET P.1, ISBN 978-1-84919-729-8.

[56] S.Fugkeaw, “Achieving privacy and security in multi-owner data outsourcing”, 2012 Seventh International Conference on Digital Information Management, Macau Aug.2012, IEEE P.239, ISBN 978-1-4673-2428-1.

[57] G.Durga Priya, S.Prathibha, “Assuring correctness for securing outsourced data repository in cloud environment”, 2014 International Conference on Advanced Communication Control and Computing Technologies, Ramanathapuram May.2014, IEEE P.1745, ISBN 978-1-4799-3913-8.

[58] C.Wang, N.Cao, K.Ren, W.J.Lou, “Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data”, IEEE Transactions on Parallel and Distributed Systems, Volume 23, Issue 8, Dec.2011, IEEE P.1467, ISSN 1045-9219.

[59] Z.Raghebi, M.R.Hashemi, “A new trust evaluation method based on reliability of customer feedback for cloud computing”, 2013 10th International ISC Conference on Information Security and Cryptology, Yazd Aug.2013, IEEE P.1, DOI 10.1109/ISCISC.2013.6767353.

[60] T.Ristenpart, E.Tromer, H.Shacham, S.Savage “Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds”, in Proceedings of the 16th ACM Conference on Computer and Communications security, New York 2009, ACM P.199, ISBN 978-1-60558-894-0.

[61] M.Factor, D.Hadas, A.Hamama, N.Har’el, E.K.Kolodner, A.Kurmus, A.Shulman-Peleg, A.Sorniotti, “Secure Logical Isolation for Multi-tenancy in cloud storage”, 2013 IEEE Symposium on Mass Storage Systems and Technologies, Long Beach May.2013, IEEE P.1, ISBN 978-1-4799-0217-0

[62] T.C.Yang, M.H.Guo, “An A-RBAC mechanism for a multi-tenancy cloud environment”, 2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronics Systems, Aalborg May.2014, P.1, ISBN 978-1-4799-4626-6.

[63] X.Y.Li, Y.Shi, Y.Guo, W.Ma, “Multi-Tenancy Based Access Control in Cloud”, 2010 International Conference on Computational Intelligence and Software Engineering, Wuhan Dec.2010, IEEE P.1, ISBN 978-1-4244-5392-4.

[64] B.Tang, Q.Li, R.Sandhu, “A multi-tenant RBAC model for collaborative cloud services”, 2013 Eleventh Annual International Conference on Privacy, Security and Trust, Tarragona, July.2013, IEEE P.229, DOI 10.1109/PST.2013.6596058.

[65] J.Wang, Y.Zhao, S.Jiang, J.J.Le, “Providing privacy preserving in Cloud computing”, 2010 3rd Conference on Human System Interactions, Rzeszow May.2010, IEEE P.472, ISBN 978-1-4244-7560-5.

[66] P.Yang, X.L.Gui, J.An, J.Yao, J.C.Lin, F.Tian, “A retrievable data perturbation method used in privacy-preserving in cloud computing”, Communications, Volume 11, Issue 8, Aug.2014, IEEE P.1637, ISSN 1673-5447.

[67] S.Haas, S.Wohlgemuth, I.Echizen, N.Sonehara, G.Muller, “Aspects of privacy for electronic health records”, Int J Med Inform, Volume 80, Number 2, Feb.2011, Epub P.26, DOI 10.1016/j.ijmedinf.2010.10.001.

[68] C.Wang, Q.Wang, K.Ren, W.J.Lou, “Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing”, 2010 IEEE Proceedings of INFOCOM, San Diego March.2010, IEEE P.1, ISBN 978-1-4244-5836-3.

[69] C.Wang, N.Cao, K.Ren, W.J.Lou, “Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data”, IEEE Transaction on Parallel and Distributed Systems, Volume 23, Issue 8, Dec.2011, IEEE P.1467, ISSN 1045-9219.

[70] C.X.Leng, H.Q.Yu, J.M.Wang, J.H.Huang, “Securing personal health records in the cloud by enforcing sticky policies”, TELKOMNIKA Indonesian Journal of Electrical Engineering, Volume 11, Number 4, 2013, P.2200, DOI 10.11591/telkomnika.v11i4.2406.

[71] S.Narayan, M.Gagne, R.Safavi-Naini, “Privacy preserving EHR system using attribute-based infrastructure”, in Proceeding of 2010 ACM workshop on Cloud Computing Security, New York 2010, ACM P.47, ISBN 978-1-4503-0089-6.

[72] S.Pearson, “Taking account of privacy when designing cloud computing services”, in Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, Washington 2009, IEEE P.44, ISBN 978-1-4244-3713-9.

[73] D.Boneh, M.K.Franklin, “Identity based encryption from the Weil pairing,” in Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, London 2001, Springer P.213, ISBN 3-540-42456-3

[74] B. Lynn, June 2007, “On the implementation of pairing-based cryptosystems,” PhD Thesis, Stanford University, 126pp.

[75] D.Boneh, B.Lynn, H.Shacham, “Short signatures from the Weil pairing”, in Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, London 2001, Springer P.514, ISBN 3-540-42987-5.

[76] M.Blaze, G.Bleumer, M.Strauss, “Divertible Protocols and Atomic Proxy Cryptography”, Advances in Cryptology-EUROCRYPTO’98, International Conference on the Theory and Application of Cryptographic Techniques, Finland May 1998, Springer P.127, ISBN 978-3-540-69795-4.

[77] G.Ateniese, K.Fu, M.Green, S.Hohenberger, “Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage”, ACM Transactions on Information and System Security, Feb 2006, Volume 9, Issue 1, ACM P.1, DOI 10.1145/1127345.1127346.

[78] Z.Yan, X.Y.Li, M.J.Wang and A.V.Vasilakos, “Flexible Data Access Control based on Trust and Reputation in Cloud Computing”, IEEE Transactions on Cloud Computing, Submitted.

[79] PBC Library Manual, http://crypto.stanford.edu/pbc/manual.html#_type_a_internals.

[80] M.Li, S.C.Yu, Y.Zheng, K.Ren, W.J.Lou, “Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption”, IEEE Transactions on Parallel and Distributed Systems, January 2013, Volume 24, Issue 1, IEEE P.131, ISSN 1045-9219.

Appendix: Publication List and Claim of Contribution

1. **X.Y.Li**, Z.Yan, P.Zhang, “A Review on Privacy-Preserving Data Mining”, 2014 IEEE International Conference on Computer and Information Technology, Xi’an Sept.2014, IEEE P.769, DIO 10.1109/CIT.2014.135.

   Contribution: I contributed to a survey of major techniques for privacy preserving data mining. I classified and analyzed the major techniques in terms of their pros and cons, user scenarios, and discussed some open issues in this area.

2. Z.Yan, **X.Y.Li**, R.Kantola, “Controlling Cloud Data Access Based on Reputation", Mobile Networks and Applications, Springer, March 2015, Springer, ISSN 1572-8153.

   Contribution: I contributed to the implementation and performance evaluation of the data access control scheme based on reputation and PRE.

3. Z.Yan, **X.Y.Li**, R.Kantola, “Personal Data Access Based on Trust Assessment in Mobile Social Networking”, 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, Beijing Sept.2014, IEEE P.989, DOI 10.1109/TrustCom.2014.131.

   Contribution: I contributed to the implementation and performance evaluation of the data access control scheme based on Individual Trust and CP-ABE.

4. Z.Yan, **X.Y.Li**, M.J.Wang and A.V.Vasilakos, “Flexible Data Access Control based on Trust and Reputation in Cloud Computing”, IEEE Transactions on Cloud Computing, Submitted.

   Contribution: I contributed to the implementation and performance evaluation of the heterogeneous data access control scheme.